Value Webs: Using Ontologies to Bundle Real-World Services

Hans Akkermans, Ziv Baida, and Jaap Gordijn, *Free University Amsterdam*

Nieves Peña, Ander Altuna, and Iñaki Laresgoiti, *Labein*

The OBELIX project has developed a component-based ontology for real-world services, along with methods and tools for graphical modeling of services and for knowledge-based configuration of service bundles.

Real-world services—that is, non-software-based services—differ significantly from Web Services, usually defined as software functionality accessible and configurable over the Web. Because of the economic, social, and business importance of the service concept in general, we believe it's necessary to rethink what this concept means in an ontological and computational sense. Most current semantic approaches focus on the computer science and software aspects of services insofar as they lie within the Web environment. Such a view is incomplete and unnecessarily limiting, because current Web Services ignore significant elements of the service context. A Semantic Web approach can—and should in our view—represent and reason about what we call *value webs*: services that have a grounding in the real world, beyond their Web elements and references alone.

The OBELIX (*O*ntology-*B*ased *El*ectronic *I*ntegration of Comple*x* Products and Value Chains) project has therefore developed a generic component-based ontology for real-world services. This OBELIX service ontology is first of all a formalization of concepts that represent the consensus in the business science literature on service management and marketing. In addition, our ontology embodies a number of systems-theoretic notions that specify how diverse *service elements*, seen as individual "Lego blocks," can connect to each other to form a larger service system: a *service bundle*. Here, various topological connection and typing rules play a key role, similar to the design of complex engineering systems.

Furthermore, we express our service ontology in a graphical, network-style representation, and we've developed support tools that facilitate end-user modeling of services. Then, automated knowledge-based configuration methods let business designers and analysts analyze service bundles. We've tested our ontology, methods, and tools on applications in real-world case studies of different industry sectors.

An interdisciplinary approach

We believe that ontology research and application can cover and handle more than current semantic approaches do. First, Semantic Web Services research still lacks convincing realistic industrial use cases that are the basis of practical use and empirical validation of ontologies and problem-solving methods.

Second, ontology research and application requires an interdisciplinary approach to services that draws not only from computer science and AI but also from economics, systems theory, and business practice. Such an approach will lead to ontological descriptions that capture much richer service profiles, even of hitherto unseen services. For example, the need exists to more clearly distinguish requester- (customer) and supplier-oriented service descriptions. Suppliers and customers have different roles in, and viewpoints on, a service, which leads to different ontological commitments and descriptions. Furthermore, service industry practice shows that "nonfunctional" aspects (such as quality features) are empirically important to service selection, composition, marketing, and sales.

Third, implementation of richer service descriptions requires Semantic Web reasoning methods beyond description or frame logics or process-flow-oriented algorithms such as AI-style planning.

Accordingly, this article gives a brief overview of what a broader interdisciplinary approach brings to the semantics of services, and outlines its application to some novel industrial use cases, which currently also have no working solutions through mainstream nonsemantic methodologies.

1541-1672/04/$20.00 © 2004 IEEE
Published by the IEEE Computer Society

The OBELIX Service Ontology

Service is by now a rather overloaded term. Until recently, research on services was the domain of business schools, which since the late '70s have produced a wealth of literature on service marketing and management. This literature gives a general framework on what (real world) services are and how they're different from physical goods. Widely used recent texts are *Services Marketing: People, Technology, Strategy*[1] and *Service Management and Marketing: A Customer Relationship Management Approach*.[2] An important observation for ontology research is that this literature now shows a consensus on many important points. Representative definitions of service often contain recurring elements:

- Valarie Zeithaml and Mary Jo Bitner—"Services are deeds, processes and performances."[1]
- Irving Kotler—"Any act or performance that one party can offer to another that is essentially intangible."[1]
- Christian Grönroos—"Activities … of a more or less intangible nature that normally … take place in interactions between the customer and service employees and/or physical resources or goods and/or systems of the service provider, which are provided as solutions to customer problems."[2]

Service viewpoints

The literature we just discussed has provided input material for our ontological description of real-world services. Our OBELIX service ontology distinguishes three interrelated top-level viewpoints (see Figure A):

- The *service value* viewpoint describes the service from a customer's perspective.
- The *service offering* viewpoint describes the service from a supplier's perspective.
- The *service process* viewpoint describes how the service offering is put into operation.

The service value and offering viewpoints

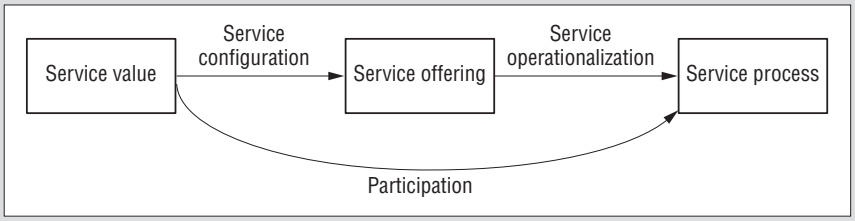


Figure A. The OBELIX service ontology distinguishes three top-level viewpoints.

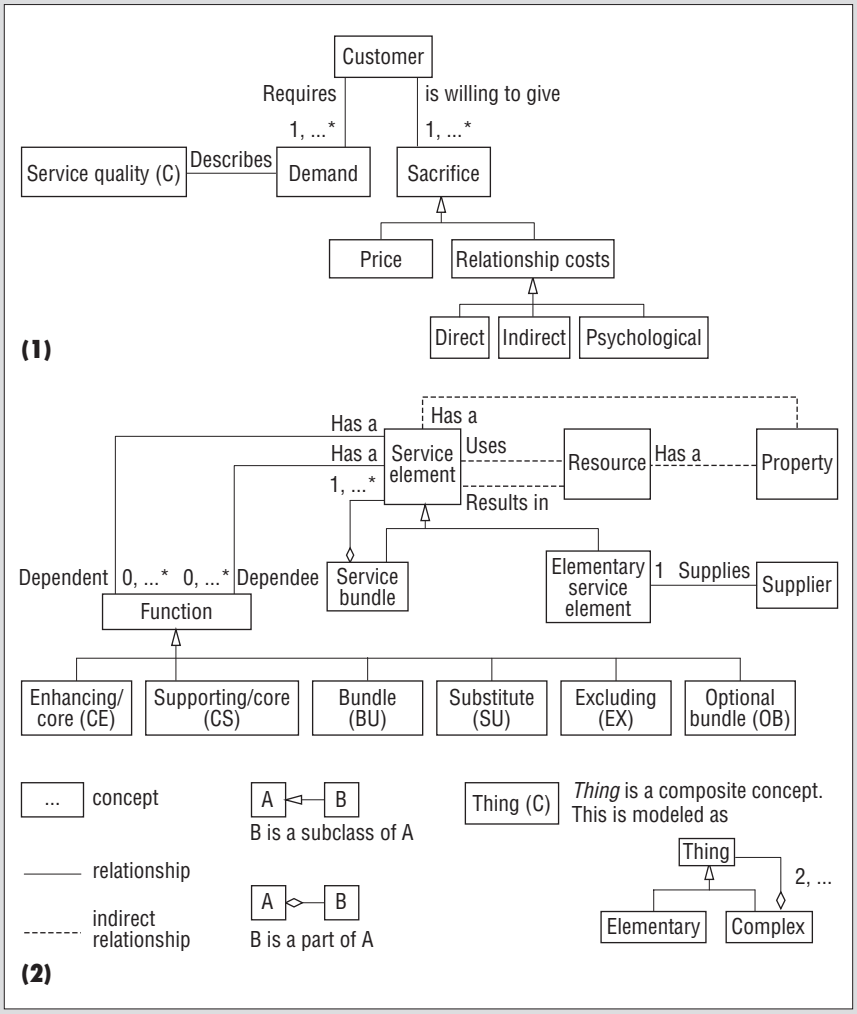


Figure B. Service subontologies representing the (1) service value (customer) viewpoint and (2) service-offering (supplier) viewpoint.

Service bundling

All industry sectors that face increasing market pressures or technological innovation face the issue of service bundling. Companies don't usually offer just a single service to a customer but a more or less interrelated collection of them. This enables broader, better coverage of customers' needs while achieving scale and scope efficiencies in service cost by sharing and reusing service elements.

To handle service bundles, we must deal with two simultaneous dimensions of complexity. First, real-world services have high variability, in that they consist of many diverse elements. In the trading of commodity goods, which still dominates e-commerce, a componential approach is already well developed (witness, for example, companies such as Dell and Cisco). In the service sector, component-based support for bundling services is still inadequate. A likely reason is

extend our *e³-value* business model ontology, which provides an ontological, graphical approach to networked business modeling. (For more on e³-value, see the "Supporting business applications" section in the main article.)

The service value viewpoint, which IT approaches often overlook, expresses the customer needs or demands that should be satisfied by acquiring a service of a certain quality, in return for a certain sacrifice. Figure B1 shows an object model of this subontology's major concepts. This subontology models service quality by employing quality frameworks used in business (for example, the SERVQUAL model). Quality features are important in profiling and composing services because they serve as selection handles for customers. One example is star ratings for hotels. Quality features also occur in e-commerce: you can download low-resolution digital photographs for free, but you must pay for high-resolution photographs. Sacrifice includes not only price but also intangible relationship costs such as customer effort spent in service coproduction, and inconvenience (waiting in line, and search and download time).

The service-offering viewpoint (see Figure B2) represents the supply-side perspective of a service. This perspective centers around the concept of a *service element*. Figure B2 shows only the ontological elements necessary to understand the business nature of this perspective. Service elements represent what a supplier offers to its customers. They're what the business literature defines as a service—a business performance of a typically intangible nature. Examples are money transfer, transportation, haircuts, Internet radio, and electricity supply. A service element can be decomposed into smaller service elements, as long as these smaller elements can be offered to customers separately, possibly by different suppliers.

The business literature distinguishes different roles or functions of service elements from a supplier perspective:

- A core service (the main business)
- A supporting supplementary service (it makes the core service possible)
- An enhancing supplementary service (it improves the core service value by adding features)

Service elements might also be substitutes for one another or exclude one another.

These functional rules concerning individual service elements are important in computing feasible *service bundles*. A service bundle is a set of service elements that can be provisioned together as a whole that's compatible with these rules.

We don't discuss the service process viewpoint here because the literature already contains much research that's adoptable (for example, ebXML [Electronic Business Using Extensible Markup Language], Web Services Flow Language, Business Process Execution Language, the DAML/OWL-S service process,[3] or Petri net semantics[4]); our work focuses more on service profiles. However, one concept that's key to both the service-offering and service process viewpoints is that of *resource*: anything that's either an input or an outcome of a service element.

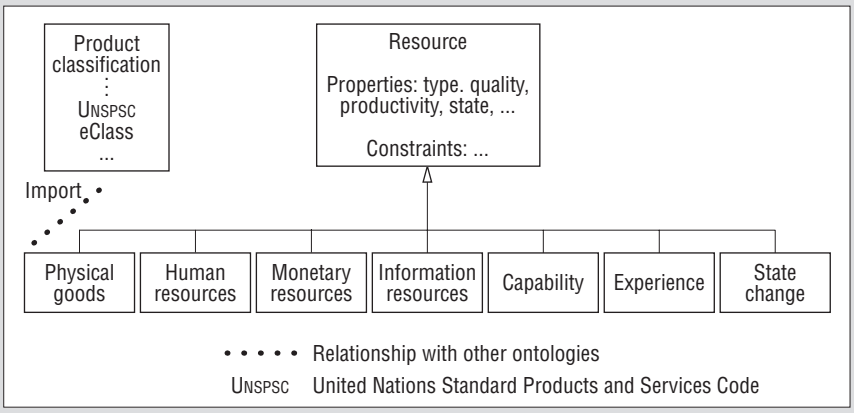


Figure C. Resources are typed inputs or outcomes of real-world service elements.

Typing

In real-world services, resources are typed (see Figure C). (This typing is absent in the DAML/OWL-S notion of resources.) These types clearly show services' great variability. Some services might result, at least partially, in tangible results (for example, a meal in a restaurant), whereas other service outcomes might be highly intangible. A theater or art museum aims to produce a memorable customer experience. A haircut, a trip, or car maintenance basically results in a state change. Many services produce an even more abstract outcome—a capability or right to do something: a credit card, formal education, a driver's license, or clearing music rights. This typing of resources plays an important role in our automatic configuration of services (see the sidebar "The Configuration Algorithm"), because it provides additional constraints on the possible outcome-input connections between different service elements.

References

1. C. Lovelock, *Services Marketing: People, Technology, Strategy*, 4th ed., Prentice Hall, 2001.
2. C. Grönroos, *Service Management and Marketing: A Customer Relationship Management Approach*, 2nd ed., John Wiley & Sons, 2000.
3. S. McIlraith, T. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001, pp. 46–53.
4. W. van der Aalst, "Don't Go with the Flow: Web Services Composition Standards Exposed," *IEEE Intelligent Systems*, vol. 18, no. 1, 2003, pp. 72–76.

that the high variability and intangibility of real-world service elements easily escapes automated description and design.

Second, many services are coproductions of different suppliers. A commonly quoted aspect of services is that customers themselves often take part in their production. For instance, many fast-food outlets suppose that their customers clean the tables. Another common example is Internet service provision, in which the customer must have both IP facilities and telecommunications connectivity. These resources usually come from different companies, although most customers perceive Internet service provision as a single, undivided service. So, we must be able to model a coproduction network of component-based service supply.

Service bundling is different from Web Service composition. The latter currently occurs at the workflow or service process

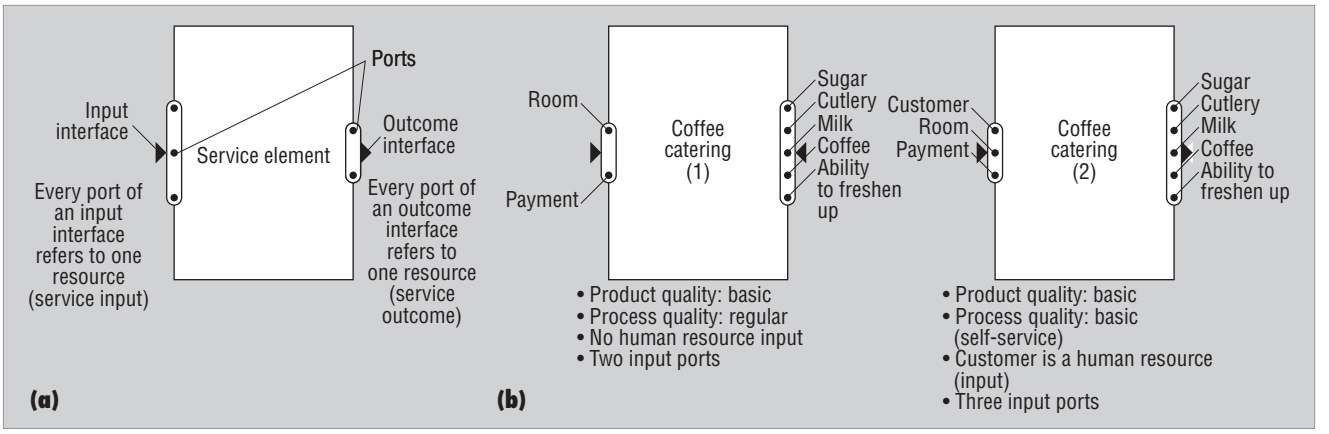


Figure 1. Service elements: (a) a generic service element; (b) different instances of the same service element in a bundle for organizing a conference. In the first instance, service personnel pour coffee; in the second, conference attendees pour their own coffee.

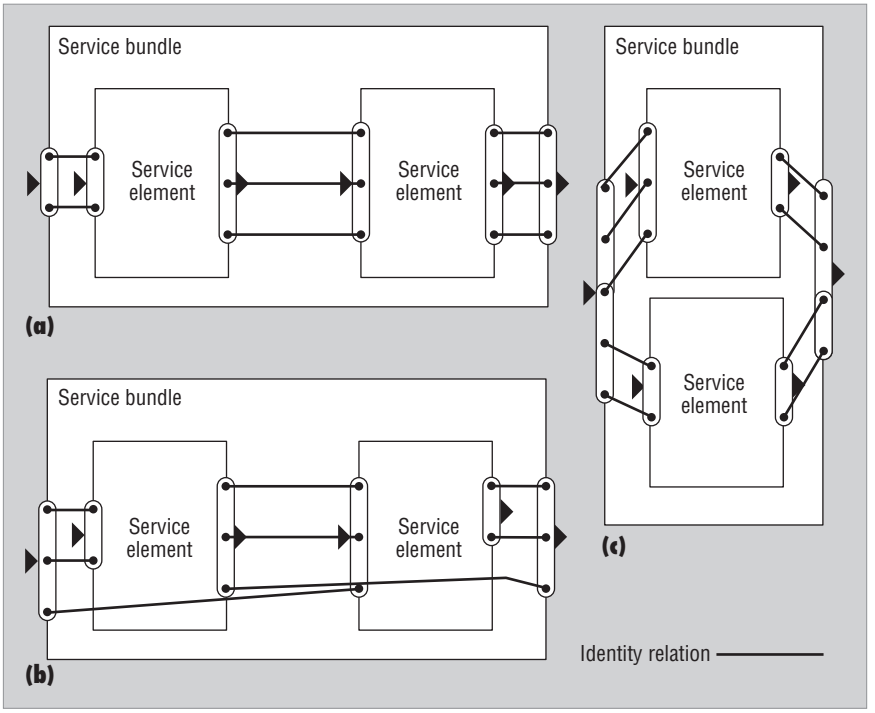


Figure 2. Different kinds of service bundling: (a) strongly connected service elements; (b) weakly connected service elements; (c) independently connected service elements.

level. Associated computational techniques such as planning thus incorporate ontological commitments to temporal notions such as states and their ordering. In contrast, service bundling is a business development activity that lives at the service profile level (to use the DAML/OWL-S terminology). Its declarative specification makes commitments not in terms of time and flow, but only to mereological and topological notions such as connectivity. This leads to the need for other reasoning methods. For example, in this article we describe configuration techniques that are adaptations to the service domain of problem-solving methods originating from the design of large-scale technical systems.

Graphical modeling and configuration

We developed the OBELIX service ontology (see the related sidebar) using ontology editors such as OntoEdit and Protégé, and the ontology is available in RDF Schema form. Testing in industrial use cases from very different sectors is the main empirical mechanism for ontology validation. However, not many practitioners can do their work directly from representations such as RDF. So, we've developed graphical representations for services and their composition that are much more intuitive for domain experts and practitioners. These graphical techniques help model services in a configurable way.

Visualizing service elements

Figure 1a shows the visualization of a service element in our ontology. We visualize service elements similarly to engineering-system components, so that we can use the knowledge-based configuration methods to automatically create service bundles. Configuration is a well-researched design task, simplified by the availability of a set of predefined components, connections, and associated parameters and constraints.[1,2]

Ports model a component's possible connectivity with other system parts.[3] In technical systems, ports are often typed: a wall outlet actually represents ports for making electrical connections. In service modeling, we use analogous structuring ideas. Every service element has two types of ports: *input ports* and *outcome ports*. The provisioning of any service element requires *resources* (inputs or outcomes of a service element) and results in the availability of other resources. An input port indicates a certain resource that's a prerequisite for carrying out this service element; an outcome port indicates the result of carrying out this service element. A special characteristic of our service ontology is that resources are typed (see the sidebar, "The OBELIX Service Ontology"). Each resource has a corresponding port. Figure 1b shows two instances of the same service element for organizing a conference.

A service element can be provisioned only if all required inputs are available and if the provisioning results in the availability of all outcome resources. The set of all input ports forms the element's *input interface*; the set of all outcome ports forms its *outcome interface*.

Knowledge-based configuration of service bundles also requires *constraints*. We distinguish two main types of constraints. *Inherent constraints* are on service-related values (such as quality properties of associated resources or conditional input-outcome constraints on the relation between quality level and payment). *Functions* are constraints on the role dependencies between service elements. In this way, we've achieved an ontological and visual description of service elements as configurable components in a larger service bundle system.

Constructing service bundles

Figure 2 depicts different ways to construct a service bundle from several service elements. As a composite service element, a service bundle also has an input and an outcome interface. These interfaces are identical to the union of all the input and outcome interfaces of that bundle's service elements.

Two exceptions to this general rule exist. First, certain resources can be *consumable* more than once; that is, they can appear in several service elements (particularly, **information** resources can be used multiple times).

Second, for resources that have the *compositeness* property, we can model multiple resources of the same type as a single resource. For instance, when two bundled service elements both require a **payment** input, we can compose a single **payment** resource from these two inputs. Often, the price for such bundling is lower than the sum of the separate prices.

A service bundle's input interface must provide all the inputs of all that bundle's service elements, unless they're provided internally (one service element might produce an outcome that a different service element consumes as an input). In Figure 2, connection links between ports mean that one port uses a resource that another port provides.

Service bundling is recursive: any service bundle is a service element. Figure 3 gives a small example for online organizing of conferences. This glass-box view shows a service bundle's internal service element structure and is of particular interest to service suppliers. The associated black-box view (not shown here) ignores the internal structure; it contains only the service bundle's inputs and outcomes as a whole. So, it's often the most useful view for letting the end customer know whether the services satisfy his or her external requirements. The combination thus yields a rich approach to service profiling with our OBELIX ontology.

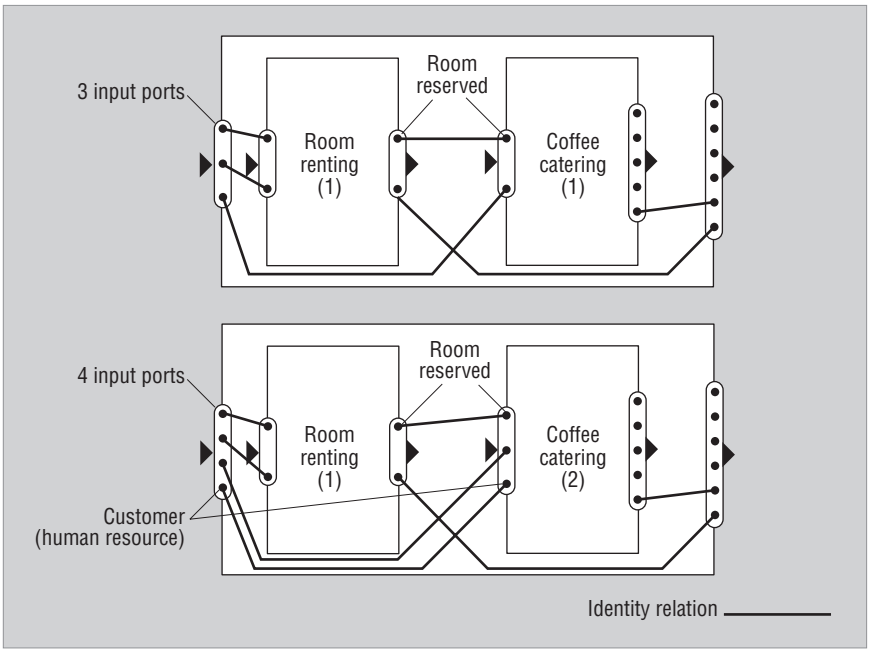


Figure 3. A glass-box view of a small service bundle for online organizing of conferences.

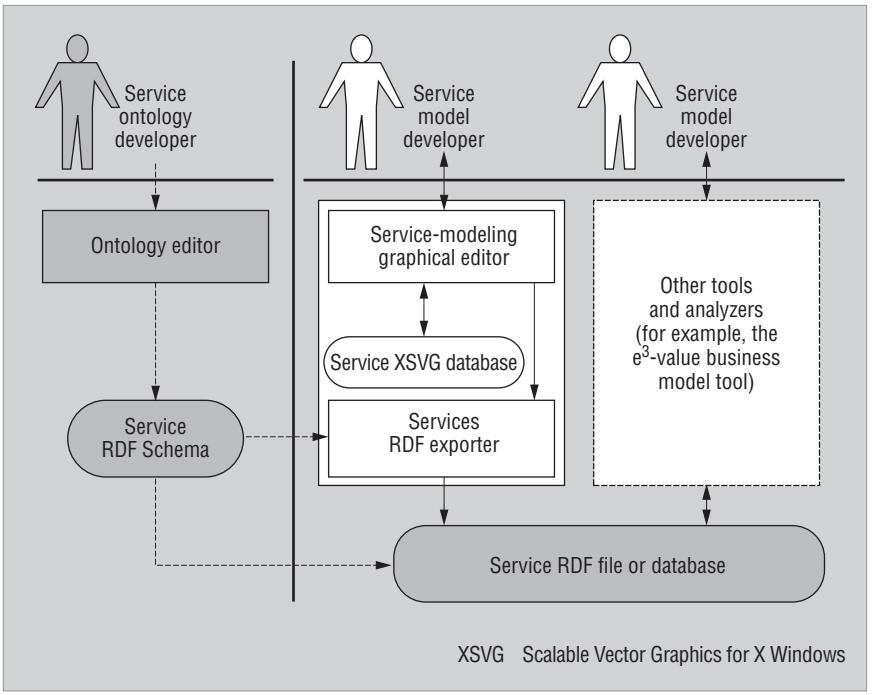


Figure 4. The architecture of the OBELIX service tool, which enables graphical modeling of services.

The OBELIX service tool

Figure 4 diagrams the architecture of the OBELIX service tool, which is a CASE (computer-assisted software engineering) tool that enables graphical modeling of services. The tool contains the underlying business rules from the OBELIX service ontology so

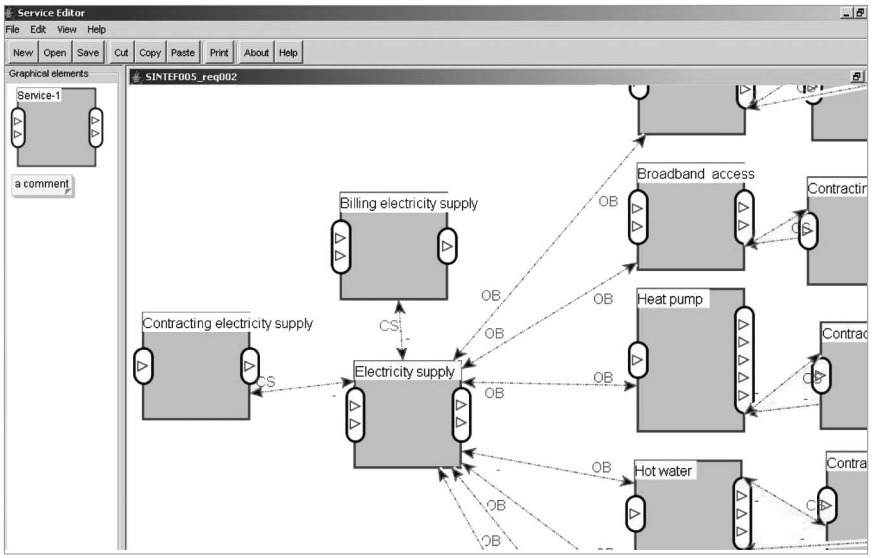


Figure 5. A screen shot from the OBELIX service tool for bundling energy services. CS means that a service element is a necessary part of the bundle in relation to the core service. OB means that it's an optional feature in a bundle.

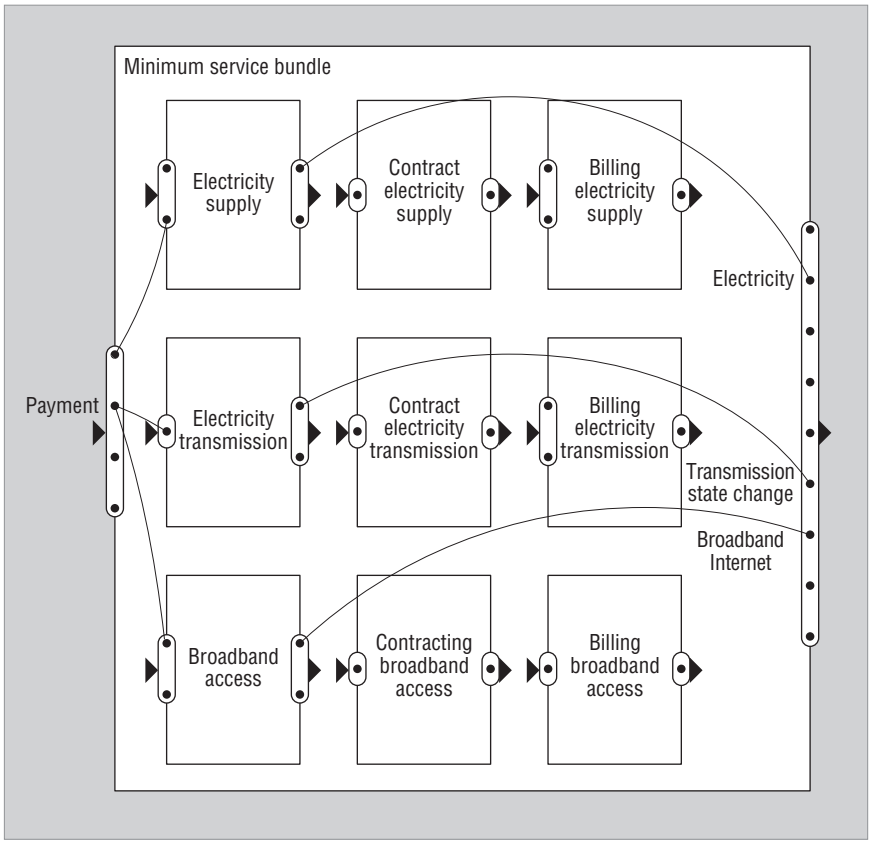


Figure 6. A computed service bundle including electricity supply (the core service) and broadband Internet access.

that it can notify users about any omissions or mistakes when they violate ontological rules. It also automatically generates the RDF representation of the service domain ontologies from the graphical model. Our knowledge-based *configuration tool* (which we discuss in the next section) employs this representation.

Supporting business applications

Using our techniques, we modeled numerous services that can be offered to customers in a bundle in various industrial case studies. One of these studies was a business analysis of service bundling related to energy supply, carried out for a Norwegian power distribution utility. In this case, individual service elements include electricity supply, electricity transmission, hot water distribution (for room heating and tap water), broadband Internet access, IT services, sales and installation of electrical appliances (heat pumps and energy control systems, to reduce energy consumption and regulate temperature), and remote-control services.

Figure 5 shows a screen shot of the service tool for this case study, indicating the dependencies between service elements. CS means that a service element is a necessary part of the bundle in relation to the core service. OB means that it's an optional feature in a bundle. EX (not shown in the figure) means that two service elements exclude one another and so can't be part of the same bundle. The screen shot shows a draft model that serves as input to our configuration tool. Other figures in this article (particularly Figures 3 and 6) show model visualizations of the situation after configuration.

Different business considerations drove this case study. Electricity is a common mass commodity in a competitive market (power markets are liberalized worldwide; Norway did so in 1991). For a power company to differentiate itself from the competition isn't easy. Because customers can freely choose their energy supplier, one strategic option is to offer better, richer customer service to retain customers. Moreover, additional customer service should exploit cost advantages by sharing and reusing existing service elements (for example, billing or maintenance). It should also improve return on investment for the existing infrastructure by using it as the carrier for more services (asset management: for example, the fiber optics for controlling the power grid might have enough extra capacity to also support broadband Internet access). So, the primary question is, what new service bundles are financially attractive for both the customer and supplier and are technically realizable?

Once we model all individual service elements, with their mutual functional depen-

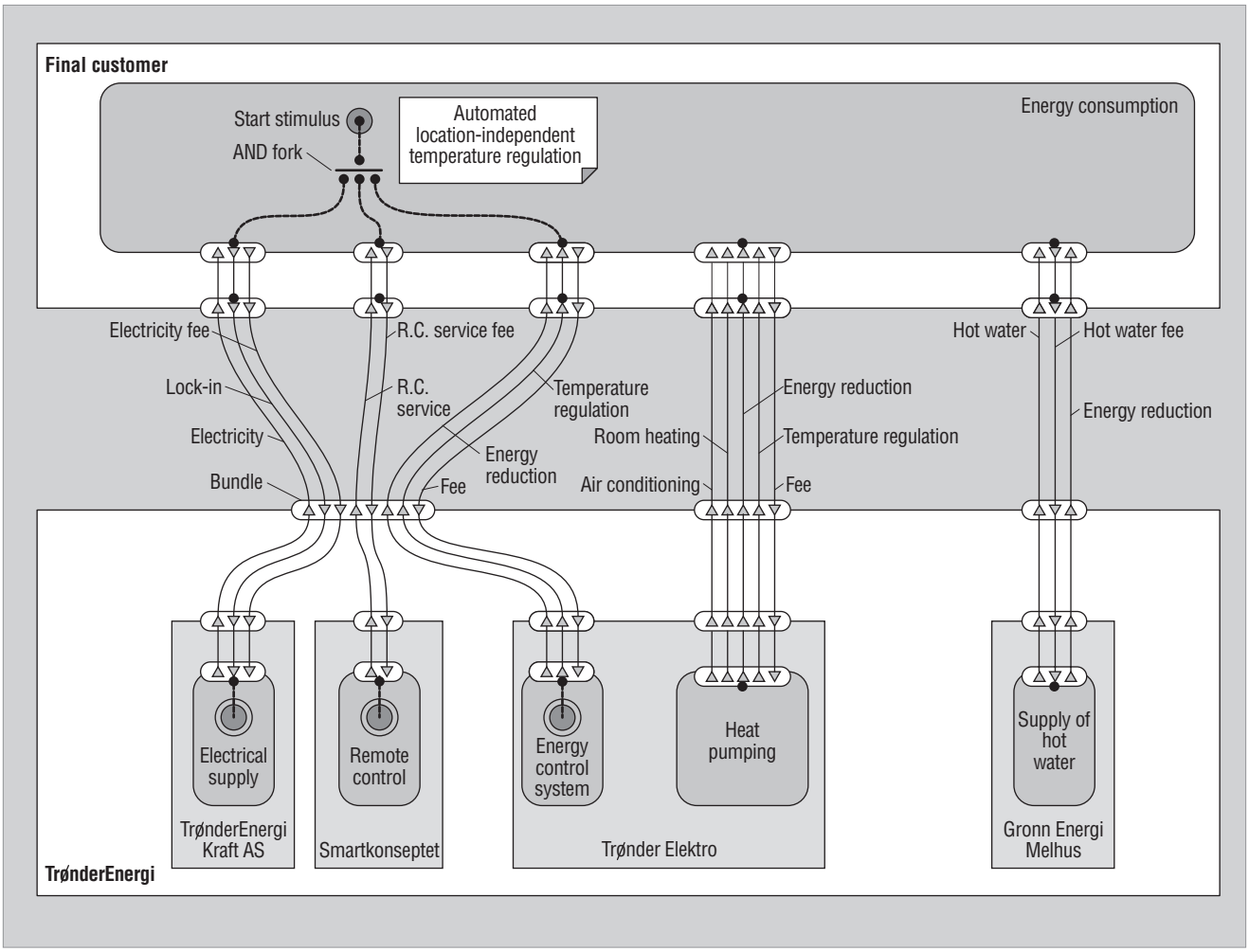


Figure 7. A possible new business model reflecting computed service-bundling decisions, as produced by our e³-value tool. This tool also helps analyze projected cash flows to assess the financial attractiveness of envisaged service bundles.

dencies, and establish the set of customer requirements, the configuration tool produces all the service bundles that are feasible with respect to all constraints and requirements. We explain our problem-solving method in the sidebar "The Configuration Algorithm." Figure 6 shows an answer that the configuration tool produced for this case.

The next step of the service business analysis is to feed the configuration results into another tool for analyzing the new business models' financial attractiveness. For this we have developed the *e³-value tool*, which is based on our ontology for networked e-business models.[4,5] Figure 7 shows a new business model reflecting service-bundling decisions that this tool produced. This modeling and simulation tool also carries out quantitative net-present-value calculations of the cash flows between all actors in spreadsheet form, to establish the profitability of new business models and service bundles. In addition, it lets users perform numerical sensitivity analyses of key parameters by computing the results of what-if scenarios.

Reflections on ontology development

The development of ontologies as formal conceptualizations is, we suggest, a key scientific method for theory formation in information science that aims to bridge human and computer understanding. We consider an ontology to be "good" if it's used in and validated against independent, external business scenarios and industrial use cases. In view of the current state of the art of Semantic Web Services, industrial-use-case testing's importance can hardly be overstated.

So, we carried out several real-world case studies in different industry sectors. Such case studies are useful in their own right and can help convince practitioners about the added value of semantics-based methods. Moreover, case studies turn out to be methodologically important as stress tests and "triangulation methods" for ontologies, reasoning techniques, and support tools. Consequently, the OBELIX ontologies have undergone significant refinement over the past years as a result of their industrial application. Specialization of a generic service ontology to a specific domain is best done by domain specialists themselves, with the generic ontology developers serving a "help desk" role.

We also noticed that for developing application-oriented ontologies, the available general tools and languages aren't very suitable. Therefore, we developed the graphical formats to enable direct visual modeling. The

The Configuration Algorithm

Figure D shows how the OBELIX ontology-based tools collaborate in service configuration.

The configuration method we use for service bundling is based on a generic configuration ontology (with these core concepts: components, connections, associations, constraints, and requirements).

First, we map the OBELIX service ontology's elements onto the configuration ontology (for more on the OBELIX service ontology, see the related sidebar). Next, the *configuration tool* computes all feasible configurations; these are then fed back to the *service tool* for visualization and further analysis.

As inputs, the configuration algorithm uses three subontologies of the configuration ontology. A *components ontology* describes components (service elements) and their associated resources (inputs and outcomes).

A *constraints ontology* defines various types of constraints. There are mainly two types. *Functions* are dependencies between service elements. *Inherent constraints* include the constraints that no loops are permitted and that an outcome port must be connected to an input port, and various other business rules stemming from the service ontology.

Finally, a *requirements ontology* describes restrictions on the desired inputs and outcomes to guide configuration. These requirements express the end customer demands by defining the type of resources required and the constraints on their property values.

The configuration tool outputs to the service tool an ontology (in RDF form) that provides all feasible service bundles such that the computed solutions

- Explicate what service elements are part of the bundle and how they're connected through the resources used
- Obey all given constraints
- Satisfy all input customer requirements

Configuration has two phases. Suppose the user has as requirements to get resources *R*1 and *R*2, and that service element *X*1 provides *R*1, and services *Y*1 and *Y*2 provide *R*2.

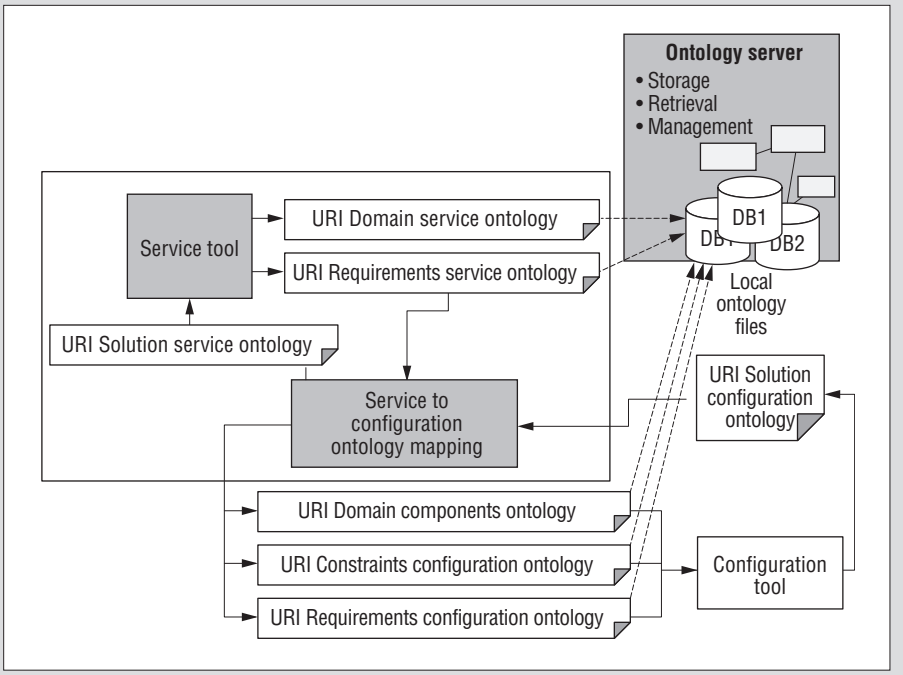


Figure D. How the OBELIX ontology tools collaborate in service configuration.

High-level configuration

This phase has two steps. The first consists of identifying initial elements for the service bundle. On the basis of the given requirements, the configuration tool knows which resources are required (*R*1, *R*2). It searches for all service elements that provide the requested resource types, and finds that *X*1 provides *R*1 and that *Y*1 and *Y*2 provide *R*2.

The second step involves applying functions. Some service elements might require or exclude others. Suppose we have the two functions Excluding(*X*1, *Y*1) and Core/enhancing(*Y*2, *Z*1). Applying the first function means that {*X*1, *Y*1} can't be in the same service bundle. The second function means that *Y*2 might—but doesn't have to—be bundled with service element *Z*1. So, we infer two possible service bundles: {*X*1, *Y*2} and {*X*1, *Y*2, *Z*1}. Each is a good solution. The algorithm applies functions to all the considered service elements in a bundle. Whenever the configuration tool adds a service element to a bundle, the algorithm applies available function information again to check the bundle's consistency.

These steps produce a set of feasible service bundles. These bundles provide the required resource types but don't yet guarantee that all values of resource properties that the customer specified are satisfied. The algorithm's next step performs this latter task.

Detail-level configuration

This phase creates connections between ports of the service elements in a bundle. It repeats for every service bundle that the first phase generated. This phase connects, as a rule of thumb, as many ports as possible within a bundle. Two ports in a bundle may be connected if they meet all these requirements:

- They belong to different service elements.
- They have different types (input or outcome).
- They require or provide the same resource.

The configuration engine then checks all restrictions regarding the range of service property values. Standard constraint solvers are adequate for this. Any input or outcome port that isn't connected to other ports in the same bundle will appear in the service bundle's input or outcome interface. Any high-level bundle might have multiple detail-level solutions.

Related Work

Our OBELIX (*O*ntology-*B*ased *El*ectronic *I*ntegration of Complex Products and Value Chains) service ontology and application research complements the DAML/OWL-S framework.[1] Overall, our research adds a needed interdisciplinary perspective. Its contributions and strengths are (purposely) at the service profile level, where we offer both business-relevant extensions important to any generic service ontology and additional computational methods such as configuration. We believe that, specifically, the value-based and recursive approach to service bundles, strong typing of resources, explicit handling of function dependencies between configurable service elements, and the computational treatment of nonfunctional properties such as quality are useful OBELIX contributions to service composition and analysis. The main computational focus of DAML/OWL-S is at the service process level, with a grounding in WSDL. However, we see no major technical obstacles in linking DAML/OWL-S's service process to the OBELIX service-profiling ontologies and methods.

WSMO/WSMF (Web Service Modeling Ontology/Framework, www.wsmo.org) is still in an early, underspecified stage. In essence, it's an extension of UPML (Unified Problem-Solving Method Description Language),[2] which in turn is a distributed, Web version of the CommonKADS knowledge modeling framework.[3] It thus has a knowledge-modeling outlook that's similar to OBELIX. The framework focuses especially on the mediators needed to connect the many different components of Semantic Web Services. WSMO also distinguishes between Web Services and real-world services, but unlike OBELIX, it doesn't implement the general concepts inherent to services in the ontology itself.

The Internet Reasoning Service[4] has many ideas similar to WSMO—among them its roots in UPML—but has been practically implemented. It caters for different Web Service tasks and problem-solving methods, of which OBELIX-style service configuration is one possible example. Domain ontologies contain static information on service applications. The OBELIX service ontology takes a significant new step here, because it provides a generic ontology that's applicable to a wide class of service domains.

Like WSMO, DOLCE[5] has a weak intrinsic conceptual notion of services. However, it provides mechanisms for further formalization and axiomatic grounding of service ontologies such as ours in terms of foundational ontologies. For example, it gives a formalization of the idea that the same object can play more than one role in a service. This can be applied to the OBELIX ontology concept of resources and further formalize aspects of the different customer-supplier viewpoints on a service.

The visual representations of service models and ontologies discussed in this article are unique to OBELIX.

References

1. S. McIlraith, T. Son, and H. Zeng, "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001, pp. 46–53.
2. B. Omelayenko et al., "UPML: Language and Tool Support for Making the Semantic Web Alive," *Spinning the Semantic Web*, D. Fensel et al., eds., MIT Press, 2003, pp. 141–170.
3. A.T. Schreiber et al., *Knowledge Engineering and Management*, MIT Press, 2000.
4. M. Crubézy et al., "Configuring Online Problem-Solving Resources with the Internet Reasoning Service," *IEEE Intelligent Systems*, vol. 18, no. 2, 2003, pp. 34–42.
5. P. Mika et al., "Foundations for Service Ontologies: Aligning OWL-S to DOLCE," *Proc. 13th World Wide Web Conf.* (WWW 2004), ACM Press, 2004, pp. 563–572.

associated ontology language representations such as RDF and OWL stay "under the hood" of the OBELIX service tool because the tool automatically creates them from the visual diagrams. This significantly lowers the barriers for practitioners.

We intend to make publicly available extensive case and ontological data on an area of investigation that we didn't discuss here. It concerns the automatic clearing of music rights for Internet radio stations. This domain contains yet another intriguing mixture of business logic considerations and technical challenges that must be simultaneously solved in a mutually consistent fashion, in a networked situation with many different actors (music rights societies, Internet radio stations, and artists and producers, in different countries). Like the power distribution use case we discussed, this provides another rich challenge and test problem for any Semantic Web and Web Service approach to real-world services.

The overview in this article is admittedly brief and thus high-level. Extensive technical reports concerning the OBELIX ontology development and the related industrial case studies, demos, and tools are available at http://obelix.e3value.com. Several other examples and case studies of our ontology-based business modeling and analysis are available at http://busmod.e3value.com. We also plan to make the e^3-value tool, service tool, and configuration tool publicly available as open-source software. For a comparison of the OBELIX research to other related research, see the "Related Work" sidebar. ■

Acknowledgments

The European Commission partially supported this work in the context of the EU-IST Project OBELIX (*O*ntology-*B*ased *El*ectronic *I*ntegration of Comple*x* Products and Value Chains, EU-IST-2001-33144). For models, tools, and data on application examples, we're indebted to Andrei Morch, Hanne Sæle, and Gunnar Salseggen (SINTEF Energy Research and TrønderEnergi, Norway); Hans-Peter Schnurr, Hans Trost, and Wolfgang Sperling (Ontoprise, Germany); Bert Hazelaar (SENA, Netherlands); and Tim Smithers, Jessica Aguado, Carlos Pedrinaci, and Amaia Bernaras (Technological Park of San Sebastian, Spain). Arthur Koks, Dennis Veltrop, Rashid Sohrabkhan, and Stephan Hoekstra (Free Univ. Amsterdam) did much of the tool development reported in this article. We also thank Terry Payne (Univ. of Southampton) for many in-depth discussions on DAML/OWL-S, and the anonymous reviewers for several useful comments that we incorporated.

References

1. T. Gruber, G. Olsen, and J. Runkel, "The Configuration Design Ontologies and the VT Elevator Domain Theory," *Int'l J. Human-Computer Studies*, vol. 44, nos. 3–4, 1996, pp. 569–598.
2. C. Löckenhoff and T. Messer, "Configuration," *The CommonKADS Library for Expertise Modeling—Reusable Problem Solving Components*, J. Breuker and W. Van De Velde, eds., IOS Press, 1994, Chap. 9.

Intelligent IEEE Systems

How to Reach Us

Writers

For detailed information on submitting articles, write for our Editorial Guidelines (isystems@computer.org) or access www.computer.org/intelligent/author.htm.

Letters to the Editor

Send letters to

Dennis Taylor, Lead Editor
IEEE Intelligent Systems
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
dtaylor@computer.org

Please provide an email address or daytime phone number with your letter.

On the Web

Access www.computer.org/intelligent for information about *IEEE Intelligent Systems*.

Subscription Change of Address

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *IEEE Intelligent Systems*.

Membership Change of Address

Send change-of-address requests for the membership directory to directory.updates@computer.org.

Missing or Damaged Copies

If you are missing an issue or you received a damaged copy, contact membership@computer.org.

Reprints of Articles

For price information or to order reprints, email isystems@computer.org or fax +1 714 821 4010.

Reprint Permission

To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at copyrights@ieee.org.

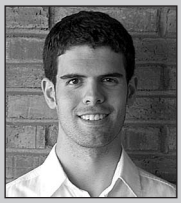
The Authors

Hans Akkermans is a professor and the head of the Business Informatics Section at the Free University Amsterdam and is an international consultant. His research interests are interdisciplinary information science, distributed intelligence, and innovation with advanced information and communication technologies (ICT). He's the chair of the board of the Netherlands Graduate Research School of Information and Knowledge Systems, the scientific director of EnerSearch AB, an international industrial consortium on ICT in energy, and the coordinator of this area for the European Commission. He also leads Vubis, the Amsterdam multidisciplinary research center for business information sciences. He holds a cum laude PhD in theoretical physics from the University of Groningen. Contact him at Vrije Universiteit Amsterdam, Faculty of Sciences, Section Business Informatics FEW/BI, De Boelelaan 1081a, NL-1081 HV Amsterdam, Netherlands; hans.akkermans@akmc.nl.

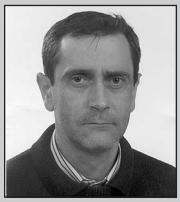
Ziv Baida is a researcher in the Business Informatics Section at the Free University Amsterdam. He's developing the Obelix service ontologies and tools, and some Obelix industrial use cases. He received his MSc in computer science and software engineering from the Free University Amsterdam. Contact him at Vrije Universiteit Amsterdam, Faculty of Sciences, Section Business Informatics FEW/BI, De Boelelaan 1081a, NL-1081 HV Amsterdam, Netherlands; ziv@cs.vu.nl; www.cs.vu.nl/~ziv.

Jaap Gordijn is an assistant professor of e-business at the Free University Amsterdam. He previously was a consultant for Deloitte & Touche and Cisco. His research interests are networked business modeling, value-based requirements engineering, and innovative e-business applications, and he developed the e3-value ontology and related methodology and tools. He received his PhD in computer science from the Free University Amsterdam. Contact him at Vrije Universiteit Amsterdam, Faculty of Sciences, Section Business Informatics FEW/BI, De Boelelaan 1081a, NL-1081 HV Amsterdam, Netherlands; gordijn@cs.vu.nl; www.cs.vu.nl/~gordijn/research.htm.

Nieves Peña works on developing intelligent systems and semantic technologies at Labein. She previously worked for Xerox on the Mobiledoc project, which integrated Internet technology with mobile clients. She received her bachelor degree in physics (electronic and automatic) from the University of the Basque Country. Contact her at the Information Society and Regional Development Unit, Labein, Parque Tecnológico de Bizkaia, Edificio 700, 48160, Derio, Bizkaia, Spain; npena@labein.es.



Ander Altuna is a research fellow in Labein's Information and Knowledge Society Unit. His research focuses on using ontologies as knowledge bases to solve complex configuration problems. He received his MSc in electrical and computer engineering from the Faculty of Engineering of the University of the Basque Country. Contact him at the Information Society and Regional Development Unit, Labein, Parque Tecnológico de Bizkaia, Edificio 700, 48160, Derio, Bizkaia, Spain; aaltuna@labein.es.



Iñaki Laresgoiti performs research on information technologies for power systems at Labein. He coordinates the ScadaOnWeb and Obelix projects. He's the Spanish national representative at the Working Group 14 (system interfaces for distribution management) of the International Electrotechnical Commission's Technical Committee 57. He received his MSc in applied mechanics from the University of Michigan. Contact him at the Energy Unit, Labein, Parque Tecnológico de Bizkaia, Edificio 700, 48160, Derio, Bizkaia, Spain; lares@labein.es.

3. W.N. Borst, J.M. Akkermans, and J.L. Top, "Engineering Ontologies," *Int'l J. Human-Computer Studies*, vol. 46, nos. 2–3, 1997, pp. 365–406.
4. J. Gordijn and J.M. Akkermans, "Designing and Evaluating E-Business Models," *IEEE Intelligent Systems*, vol. 16, no. 4, 2001, pp. 11–17.
5. J. Gordijn and J.M. Akkermans, "Value-Based Requirements Engineering: Exploring Innovative E-Commerce Ideas," *Requirements Eng. J.*, vol. 8, no. 2, 2003, pp. 114–134.