

The SOA Magazine

Feature Article



Service Elicitation: Defining the Conceptual Service

by Cyrille Thilloy

Published: January 29, 2007 (SOA Magazine Issue IV: February 2007, Copyright © 2007)

[Download this article as a PDF document.](#)

Abstract: Fundamental to any SOA delivery project is the definition of services. More specifically, the ability to define what constitutes a service and how logic should be partitioned and represented across a collection of services. The ambitious goal of SOA to achieve unity between business and technology domains further makes service definition a critical step along a typical SOA roadmap.

This is the second article in a series dedicated to exploring the functional side of SOA. It provides several ways to describe a service as part of a stage called "service elicitation" - essentially, the process of extracting services from business knowledge.

Introduction: Service Elicitation through Formalization

It is commonly accepted that a successful SOA initiative must be business-driven rather than technology-driven. Given that perspective, the challenge for practitioners is to define services properly in order for them to be aligned with the organization's true business needs. When service definition accuracy is the goal, one needs to fully understand what the intended purpose of the service is.

Service elicitation is the process by which a service is extracted from business intelligence. The term "elicitation" is used to avoid confusion with the term "discovery" (already used to refer to specific technology-sets and the service-orientation Service Discoverability principle). The main question that service elicitation attempts to answer is "what is a service?" By knowing what a service is, it is easier to validate its accuracy.

Within the context of service elicitation, the term "service" is used as follows:

- A service is a container that can encompass collections of functions and even entire business processes.
- A service does not refer to or imply any specific implementation technology.

In fact, Erl's service-oriented analysis process [REF-3] recommends always qualifying conceptual services as "service candidates," so that they clearly communicate the fact that they do not relate to or represent a real form of implementation. In support of the business process decomposition that is part of Erl's corresponding service modeling process, even individual actions that are identified and grouped into proposed service boundaries are qualified as "service operation candidates" or "service capability candidates" for the same reasons.

During service elicitation services are commonly identified within the context of interchanges occurring as part of a "community of interest." This often involves a knowledge management type elicitation of domain knowledge about the community itself. Knowledge elicitation produces a model of the information and its flow, like the well-known mind maps or topic maps. This information is subject to transformation because knowledge of this type creates the potential for new activities or modifications of existing activities. It is understood that the domain expert may or may not have complete information regarding the overall business environment and its drivers. Thus, service elicitation is carried out iteratively and its iterative nature requires human to human involvement (Figure 1).

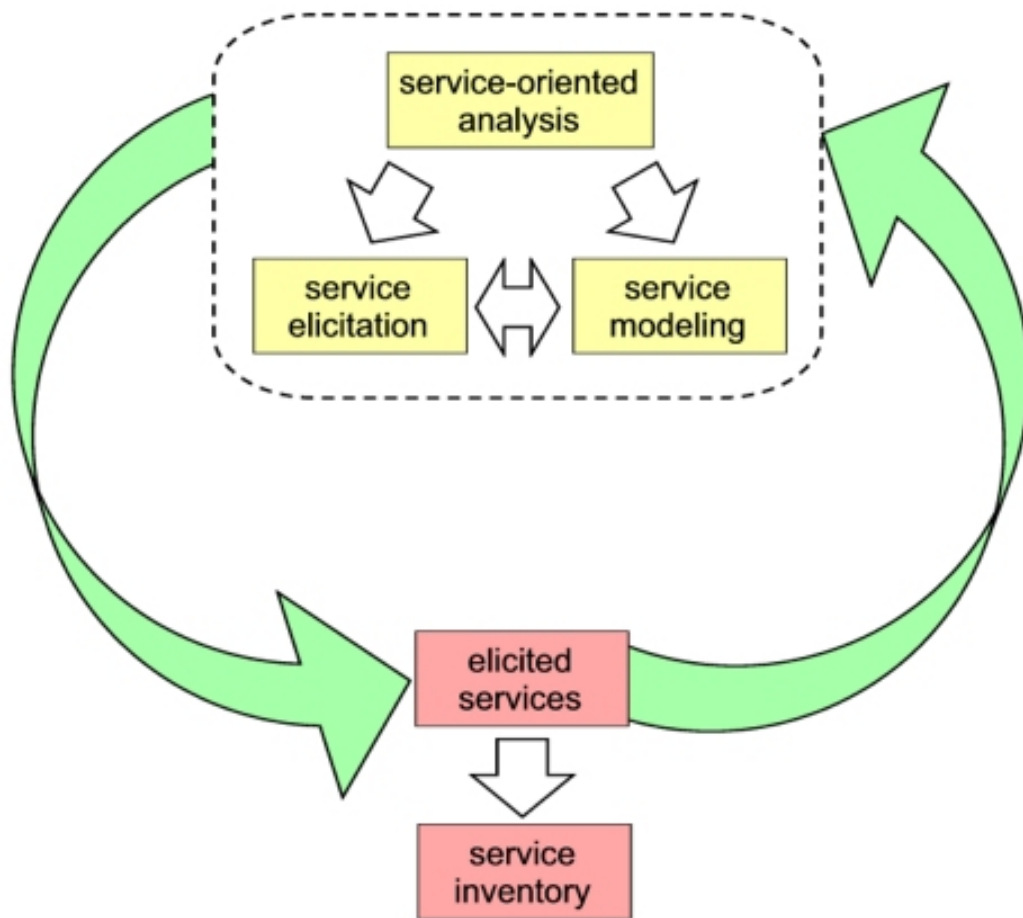


Figure 1: Service elicitation processes are typically iterative in nature, building up a service inventory through the repeated definition of new services or refinement of existing services.

New or previously hidden information structures are commonly exposed through the elicitation process. What is discovered is more than just a set of request-response exchanges already occurring in the transaction space. Elicitation may change the meaning of previously discovered information, making questions like, "given the knowledge of the information flow, what is newly possible?" necessary.

A knowledge elicitation framework, similar to the "Zachman Framework™ for enterprise architecture" [REF-12] is useful in providing completeness and familiarity within a larger methodology. However, the elicitation activity reduces the scope to three questions ("What-Who-Why" [REF-10]) at the contextual and conceptual levels of the Zachman Framework™. (The other levels and questions will be covered in the next article of this series.)

It is worth noting that service elicitation processes are not in competition with existing enterprise architecture frameworks like Zachman, TOGAF™, or others. In fact, the formalization of service information enhances these frameworks and promotes livable enterprise architecture by leveraging XML-based (and thus process-able) representation. Service elicitation simply helps produce a "conceptual blueprint" [REF-4] of a service inventory, maximizing the potential of the eventual physical service designs.

Goals	To elicit, at a high-level, what should be a service candidate.
Inputs	Business requirements, business processes, service models.
Outputs	A list of services described at a high level.

Table 1: A generic summary describing service elicitation.

Service Elicitation and Service Analysis

The definition of a service generally represents just one step (or a collection of steps) within the scope of a typical service-oriented analysis process. Prior to elicitation, information gathering steps are generally required to identify, define, or even bring up to date the necessary enterprise business models used as the basis of the analysis (and from

which the services are eventually derived).

Subsequent to initial service elicitation, there are further steps that delve deeper into the logic assigned to a given service context, so as to discover additional processing tasks required for this logic to be carried out in the real world. These discoveries frequently lead to the creation of further service candidates, and so the elicitation process repeats itself again.

Proven service-oriented analysis approaches do not limit the scope of the analysis to a single business process [REF-3]. Doing so would couple elicited services to that process, limiting their reuse potential and inhibiting their ability to be leveraged as true enterprise resources. Therefore, iteration through multiple business processes is common and ends up benefiting the elicited services in several ways:

- Prior to an actual physical service design being created, the service candidate is defined and repeatedly refined and validated by association with numerous business process requirements.
- The service candidate, being just a conceptual creation, can be significantly augmented throughout the iteration process. If there's a need to consolidate, regroup, or even split up service candidate boundaries, this is easily accomplished because, again, physical services have not yet been created.

The freedom to define and prove candidate services in relation to real world business processes allows for an opportunity to create truly robust and reusable service logic. Ultimately, this is extremely important as the final version of the service candidate forms the starting point for the service-oriented design process [REF-3] at which point a service contract is produced. All of the analysis and elicitation effort that goes into defining a service candidate ends up prolonging the lifespan of its corresponding service contract. And, as we all know, contract longevity reduces the eventual burden of SOA governance.

The part of service-oriented analysis most commonly associated with the elicitation of service candidates is that of service modeling [REF-3]. This topic goes well beyond service definition to explore the roles of services within various business contexts and scenarios and further explores service composition candidates to fulfill business process requirements. Service modeling forms the basis for the third article in this series. For now, our focus remains on the definition and formalization of service candidates.

Formal Descriptions and Classifications

The formalization of a service essentially refers to the consistent documentation of service profile information. As the elicitation process is carried out repeatedly, this information is updated and refined. At some stage it may even be ready for representation via a standardized semantic expression syntax.

The more complete and accurate the formal description, the more useful it is. However, creating a complete formal description of a service is not an easy task. It can require the involvement of key subject matter experts (such as business owners, domain experts, business analysts, process owners, or even community leaders) and the use of service classifications that label services within an enterprise context.

Most vendor SOA platforms already provide sets of "industry classifications" that establish generic categories for common service types. These service models [REF-3] essentially act as templates in that each proposes a distinct functional context. Specifically, service models define "the type of logic services encapsulate, the extent of reuse potential this logic has, and how this logic relates to existing domains within the enterprise" [REF-4].

To avoid vendor-specific terminology, we'll use the following vendor-agnostic terms ([REF-3, REF-4]):

- *Entity Service* - Functional business context associated with a business entity or a collection of related business entities. (Entity services are also known as "entity-centric business services" and "business entity services".)
- *Utility Service* - Functional non-business context associated with a related set of processing capabilities. (Utility services are also known as "application services," "infrastructure services," and "technology services".)
- *Task Service* - Functional business context associated with a specific business process. (Task services are also known as "task-centric business services" and "business process services".) A variation of the task service is the *Orchestrated Task Service* which exists within an orchestration platform that imposes specific service characteristics. (Orchestrated task services are also known as "process services," "business process services," and "orchestration services".)

For descriptions of these service models, see the sidebar. Note that these classifications are very generic and can be

further tailored to specific enterprise contexts.

XML Representation of Elicited Service Profiles

Regardless of how services are classified and documented, their descriptions can eventually be moved to processable representations like XSD, RDF, OWL [REF-9], adding the potential for dynamic interaction. By separating the data from the domain of interest (graphical representation, syntax, function, semantic), XML-based languages allow the processing of the domain of interest independently of the data. The increasing importance of XML representation and the direction it has taken toward service semantics is summarized Figure 2.

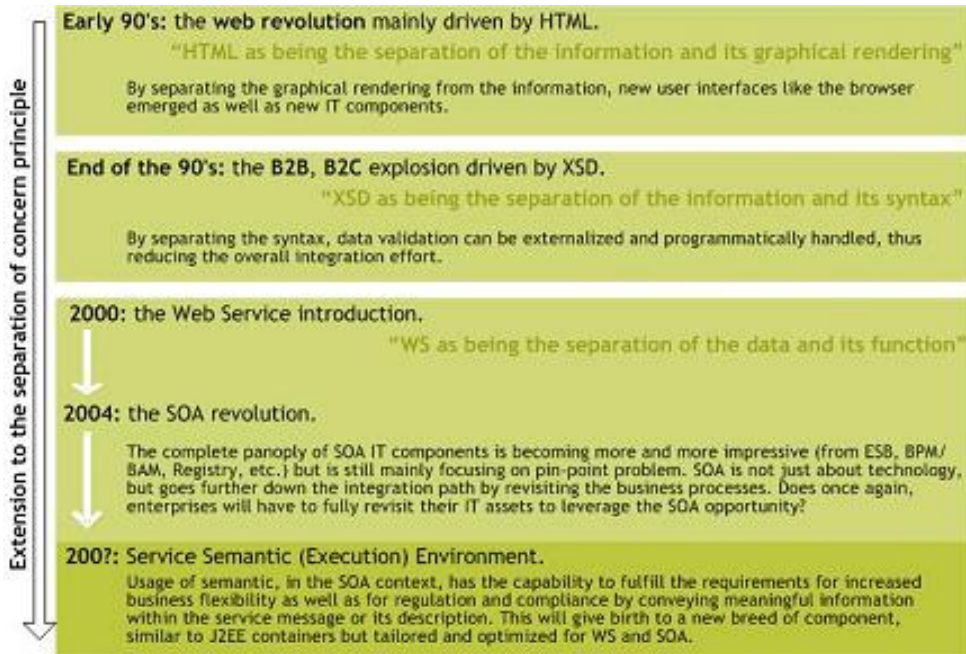


Figure 2: How the evolution of XML-based languages is working its way toward semantic service representation.

In this context the term "semantic" is distinguished from the term "metadata" in that semantic data encompasses, but is not limited to, information regarding the behavior, compliance to business rules or regulations, adherence to a model (or architecture) as well as the classification of the services.

Once available and mature in an industry-standard format, the formal semantic expression of an elicited service can be used by Web services frameworks to compose services at the technical level while the formal description of services can be used to validate this composition at the business level.

Brainstorming Techniques for Knowledge Extraction

Given that service elicitation is a form of knowledge extraction, it can benefit from existing knowledge management brainstorming techniques. For example, an approach similar to CRC [REF-8] can be used in a role game to ensure that the services are modeled correctly, as shown in Figure 3. In such a case, the focus is on the accuracy of binding functions and messages and not the accuracy of the encapsulation of data and methods into objects (a common distinction between service-orientation and object-orientation).

More about Service Models

Service models are a fundamental tool to carrying out organized and coordinated service elicitation, especially when defining larger collections of services. The service model descriptions and examples below are provided courtesy of Prentice Hall [REF-4].

Entity Services

In just about every enterprise, there will be business model documents that define the organization's relevant business entities. Examples of

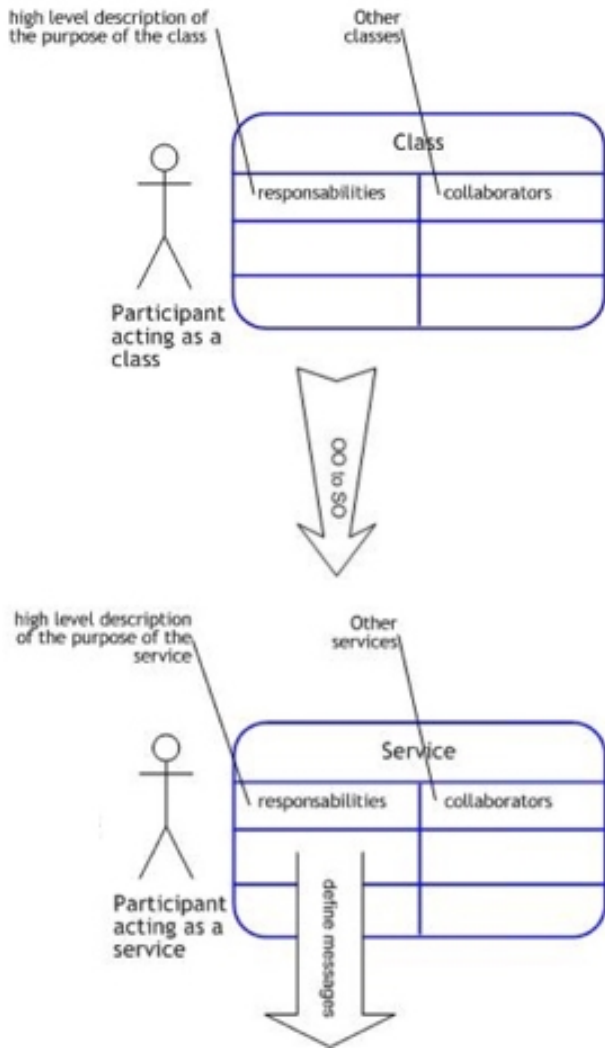


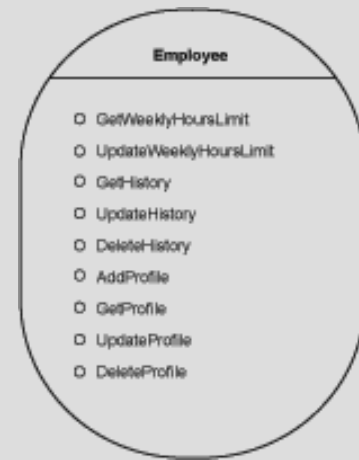
Figure 3: CRC cards used for objects and, now, services.

Alternative techniques may be helpful in identifying services and depicting information flows and service dependencies include:

- Business Process Modeling Notation (BPMN), a language usually used for process modeling and simulation, has no direct means of expressing information between the activities of a process. Therefore, when defining services, it is best suited for simulation.
- DEMO modeling forms and idioms [REF-7].
- Unified Modeling Language (UML) class diagram, sequence diagram, activity diagram or collaboration diagrams.
- Topic maps.
- Verna Allee's Value Networks™ [REF-5, REF-6].

Whatever techniques or artifact are used, they must fulfill the need of the activity: eliciting the services. The following table represents a variation of the "evolution of the organization" (originally documented in [REF-6]) providing an added context for the service-oriented enterprise.

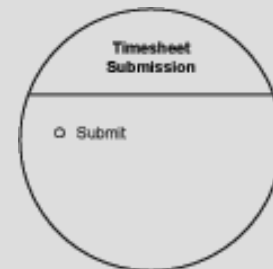
business entities include customer, employee, invoice, and claim.



The entity service model represents a business-centric service that bases its functional boundary and context on an existing, active business entity (or a collection of related business entities). It is considered a highly reusable service because it is agnostic to specific business processes and can therefore be leveraged to automate multiple processes.

Task Services

This is another type of service centered around encapsulating business logic. However, its functional boundary is directly associated with a specific parent business task or process. It therefore has less reuse potential and is generally positioned as the controller of a composition responsible for composing other, more process-agnostic services.



One point of clarification often required is in relation to entity service capabilities. Each capability essentially encapsulates business process logic in that it carries out a sequence of steps to complete a specific task. An entity Invoice

plan, organize and control	vision (goals), values and empowerment	emergence, integrity and relationships
functions	processes	services
individual tasks	work and project teams	communities
raw materials	financial capital	knowledge and intangibles

Table 2: Evolution of the organization from the early industrial age to the present [REF-6].

It is generally best for an organization to hand-pick one or more approaches that work best within its environment.

Optimization Techniques

So far the services have been described at a macro level and will need to be further refined. Just to give you a taste of some of the more advanced aspects of service elicitation, this section borrows from Mylopoulos's well known information modeling techniques [REF-1] to provide established abstraction mechanisms that can be used to further refine and even remodel services candidates and their messages.

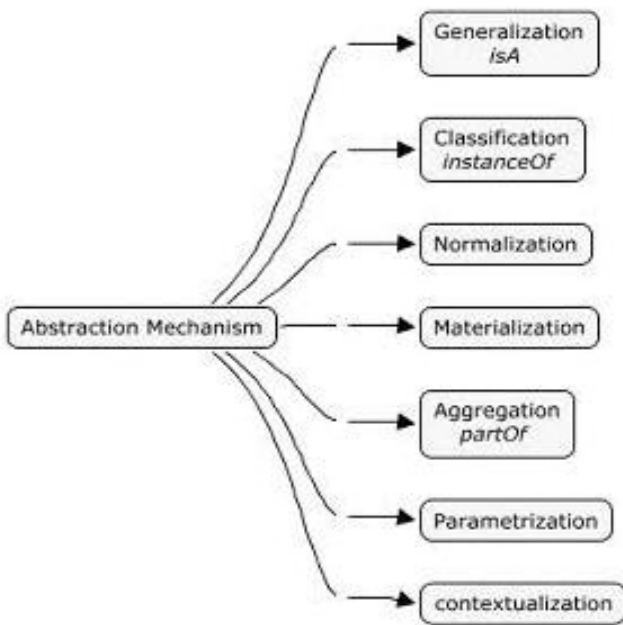


Figure 4: Common abstraction mechanisms that may also be of use for service elicitation.

Each of the abstraction mechanisms is further explained below, as it could be used to accommodate service elicitation:

- *Classification* is a fundamental abstraction mechanism sometimes referred to as the "instanceOf" process where an atom is classified under a more generic atom (the class), thereby making it an instance of the class. (Related question: "Is this element (service, function or data) an instance of another?")

Service, for example, may have an Add capability that contains process logic associated with creating a new invoice record.

How then is what a task service encapsulates different from what an entity service's capabilities contain? The primary distinction has to do with the scope of the process. The Invoice Service's Add capability is focused solely on the processing of an invoice document. If, however, we required a business process that created an invoice record, updated the corresponding purchase order and client account records, issued notifications, and adjusted inventory levels, we would have process logic that *spans* multiple entity domains. This represents a "parent" process in that it consists of processing logic that needs to coordinate the involvement of multiple services.

Services with a functional context defined by a parent business process or task can be developed as standalone Web services or components - or - they may represent an orchestration platform. In the latter case, the design characteristics of the service are somewhat distinct due to the specific nature of the underlying technology. In this case, it may be preferable to qualify the service model label accordingly. We refer to this type of service as the Orchestrated Task Service.

Utility Services

Each of the previously described service models has a very clear focus on representing business logic. However, within the realm of automation, there is not always a need to associate logic with a business model or process. In fact, it can be highly beneficial to deliberately establish a functional context that is non-business-centric. This essentially results in a distinct, technology-oriented service layer.

- *Generalization* can be seen as the "isA" mechanism that organizes objects based on their generality and specificity. (Related question: "Is this element similar (or the same) as another?")
- *Aggregation* can be seen as the "partOf" mechanism where objects aggregate their components or parts into more complex ones. (Related question: "Is this element part of another?")
- *Contextualization* allows partitioning and packaging of the descriptions (the context) being added to the information. (Related question: "Does this element differentiate from another by a specific context?")
- *Materialization* can be seen as the reverse of the generalization mechanism where a class is refined into a more concrete, materialized one. (Related question: "Is this element the materialization of another?")
- *Normalization* is a technique used when modeling becomes too complex. Only the common typical entities are modeled first and then through several passes special cases are treated (and usually extracted). (Related question: "Does this element need to be normalized?")
- *Parameterization* is used to extend the reusability of the model by adding parameters to the elements. (Related question: "Does this element differentiate from another by specific parameter(s)?")

Some of these abstraction techniques that can also be used to further refine previously defined service candidates, as follows:

- The "generalization" technique can reveal common services.
- The "parameterization" technique can reveal the need to define a facade service (using default values in front of a more generic service).
- The "aggregation" technique can reveal composable services.

Several established service-oriented analysis and service modeling processes and methodologies already incorporate these techniques. However, being familiar with them individually will better enable you to incorporate them into your own, custom service elicitation approach.

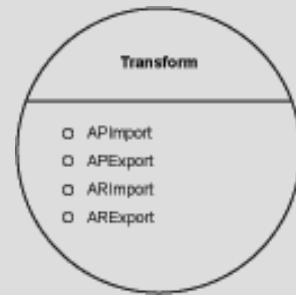
Conclusion

Service elicitation, whether carried out on its own or as part of a service-oriented analysis, produces important artifacts that carry on to form a conceptual blueprint for a given service collection or inventory. How elicitation is performed can vary, but most approaches have similar results:

- The creation of formal service documentation with an enterprise perspective.
- The validation of service definition accuracy and requirements.
- The basis for eventual dynamic processing of service semantics.

The artifacts also tie into the next stage of a service's path through the SOA delivery lifecycle: service modeling. Service modeling will be addressed in the next article of this series.

(This article contains content contributed by Multiforce technologies Inc. [REF-11])



The utility service model accomplishes this. It is dedicated to providing reusable, cross-cutting utility functionality, such as event logging, notification, and exception handling. It is ideally application agnostic, in that it can consist of a series of capabilities that draw from multiple enterprise systems and resources, while making this functionality available within a very specific processing context.

Note that entity, task, and utility service models are intentionally generic in nature in that they apply to just about any type of enterprise. Customized variations can be further derived to fulfill specific types of domain abstraction.

References

- [REF-1] "Information Modeling in the Time of the Revolution", John Mylopoulos; Information Systems 23(3-4), June 1998.
- [REF-2] "Business Centric Methodology" available at www.businesscentricmethodology.com
- [REF-3] "Service-Oriented Architecture: Concepts, Technology, and Design", Thomas Erl; Prentice Hall, 2005; ISBN-0-13-185858-0 (www.soabooks.com)
- [REF-4] "SOA: Principles of Service Design", Thomas Erl; Prentice Hall, 2007
- [REF-5] Value networks™, information available at www.vernaallee.com
- [REF-6] The Future of Knowledge: Increasing Prosperity through Value Networks, Verna Allee, Butterworth Heinemann, 2002, ISBN-0-7506-7591-8
- [REF-7] Design and Engineering Modeling for Organizations (DEMO) methodology information available at www.demonl.
- [REF-8] "A Laboratory For Teaching Object-Oriented Thinking", Kent Beck & Ward Cunningham, available at c2.com/doc/oopsla89/paper.html
- [REF-9] "A Tool for Storing OWL Using Database Technology", Maria del Mar Roldan-Garcia and Jose F. Aldana-Montes, 2005, available at www.mindswap.org/2005/OWLWorkshop/sub1.pdf
- [REF-10] "A methodology for Service Architectures", Steve Jones; Cap Gemini's contribution to OASIS Adoption Blueprints technical Committee, available at <http://blueprints.jot.com/WikiHome/Methodology+for+Service+Architectures>
- [REF-11] Enterprise Service Oriented Methodology by Multiforce Technologies, Inc. (Version 5.00, June 2006) available at <http://www.multiforce.com>
- [REF-12] Information about the Zachman Framework(tm) available at <http://www.zifa.com>.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[home](#) [past issues](#) [contributors](#) [official book site](#) [legal](#) [rss](#)

Copyright © 2006-2007 SOA Systems Inc. All Rights Reserved