A Conceptual Architecture for Pragmatic Web Services

Hans Weigand and Willem-Jan van den Heuvel
Infolab, Tilburg University
P.O.Box 90153
5000 LE Tilburg, The Netherlands
H.Weigand@uvt.nl, W.J.A.M.vdnHeuvel@uvt.nl

Abstract: In the current literature on service-oriented computing, the relationship between services and web-services is not always clear. Much research, notably in the area of service representation, discovery and composition, claims to address services whereas they actually apply to *web*-services. In this paper, we use insights from Language/Action Perspective and from value modeling to define services at an abstract (business) level. On that basis, we explore a pragmatic approach to service discovery, the cornerstone of the Service Oriented Architecture, and show how it differs from *web* service discovery. In the course of the discussion, some differences between a Semantic Web approach and a Pragmatic Web approach become apparant.

1 Introduction

Service-Oriented Architectures provide major advantages for today's enterprise information systems by presenting the interfaces that loosely coupled connections require [Pa05]. Web services [W304] seem to become the preferred implementation technology for realizing the SOA promise of service sharing and interoperability. Semantic descriptions of web services using ontologies can be exploited to improve the process of locating services that achieve a given customer goal. Much research has been devoted to the development of proper web service description languages and to the optimization of the matching process. In this context, Chris Preist has presented a useful conceptual architecture for semantic web services [Pr04]. Preist suggests a clear distinction between a service and a web service. A service is a provision of value to a client, such as a train ticket Innsbruck to Stuttgart, whereas a web service is a computational entity accessible over the Internet. A web service can be sent a SOAP message and will return a result.

The suggestion of Preist has been taken up by several researchers. According to [Fe05], the need for the distinction service/web service becomes evident if we think about a prototypical e-business scenario: a user wants to get a specific service that provides some real-value for him. Web services are technological means for accessing or specifying services offered by some specific provider. Typically, business users are not specifically interested in Web services but rather in the services. On this basis, [Fe05] argues that much work on automatic discovery of services fails to provide a viable solution as it mixes up the two concepts, and assumes wrongly that complete and correct descriptions of services are available. A proposal is elaborated for a conceptual model of service discovery in which an explicit distinction is made

between web service discovery and service discovery.
Lara and Olmedilla [LO05] also adopt the distinction of Preist. They make a distinction between web service discovery and service contracting, where web service discovery consists of two steps: first, matching client goals with an abstract service description (e.g. in description logic, [LH03]), and secondly, to check whether a concrete service can be provided by this provider. In the service contracting phase, the requester and the provider will establish a conversation in which relevant information will be exchanged (e.g. credit card data). Although the distinction between service and web service is a valid one and we subscribe to it, we observe that in many current publications both notions are still used intertwiningly. Apparently, the distinction needs to be analyzed in more detail and a more explicit theoretical basis of both concepts and their relationships is needed.
For example, is it true, as [Fe05] states, that a web service is (nothing but) a means to request a service over the Internet? And how to understand that in the proposals of [Fe05] and [LO05] web service discovery is the first and major step, whereas Preist does not talk about it at all and only talks about service discovery?
One objective of this paper is to clarify the service/web service distinction and the way in which they are connected. In particular, we will develop a theoretical basis for both concepts based on two bodies of knowledge. Firstly, we will use the Language/Action Perspective, and particularly the work of Dietz [Di06] that distinguishes between business level transactions and information-level interactions for clarifying how services and web-services are linked. Secondly, for a more precise characterization of services at the business level, we draw on recent research results in value modelling [GAV00, We06]. On the basis of this, a conceptual architecture of pragmatic web services is proposed. Some consequences for service discovery and for web service design are discussed in section 3. In the context of this paper, these issues are not worked out in depth, but only in so far they also serve the second objective of this paper: to explore what a pragmatic approach towards service discovery would mean as opposed to (or complementing) a Semantic Web approach.

2 Service versus web service

In this section we ground the difference between service and web service in two theories: the Language/Action Perspective (DEMO), and the e3-value approach. On the basis of that, we propose a conceptual architecture of pragmatic web services.

2.1 DEMO

The difference made by Preist between a web service as a computational entity, that is, as part of the Information System, and a service as a provision of value in the business domain, reflects the essential difference made in the Language/Action Perspective field between messages and what actors do with messages (speech acts) in the social or business domain. Dietz [Di06] has formalized this difference in a

"distinction axiom" that says that there are three distinct human abilities playing a role in the operation of actors: performa, informa and forma. The *forma* ability concerns the form aspects of communication and information. We are talking here about such things as uttering a sentence, or storing and retrieving documents. The *informa* ability concerns the content aspects of communication and information, where we abstract from the form aspect, as in the analysis of information needs. When we abstract from the information exchanges as well and focus on what is achieved – e.g. the creation of a commitment – then we talk about the *performa* ability. Drawing on the distinction axiom, system analysts have a way of abstracting from the *how* of communication in order to have a clear picture of the essence of the communication, that is the coordination of business activities (production acts, in DEMO terminology). From there, they can go down and find out the best way to implement the coordination acts by means of messages and documents, and during this step, Information Technology can be useful. Information systems do not perform business activities, they support them by messaging systems (implementing coordination acts) or by information processing systems (supporting production acts). Web services are software components exchanging messages and performing certain computations and inferences. They do not have performa capabilities themselves, but with their informa capabilities, they can support humans in their performa activities. Figure 1 illustrates the major distinction between the social level of human action (performa) and the information level (informa/forma) level. Agreements are made, business is done, and customers are satisfied at the social level. Information messages are exchanged at the information level, that is, by means of Information Technology or traditional media. Services have a life cycle, and the various phases of services can be supported by one or more web services. Each web service performance, irrespective of which service phase it supports, has also a life cycle, commonly called publish – find – bind [Pa05].

Let us consider a few examples. One of the services offered by a library is the loan of books. A loan is a social action: it includes the right of keeping the book for some time and reading it, and, as the right is temporary, the obligation to return the book in due time. Each party has a certain responsibility towards the other party. Essential in the coordination of their activities is coordination of the start of the loan and the ending of the loan: something changes in the social world when the book is borrowed and something changes again when the book is returned. Information systems can be used for performing a coordination act – e.g. the request to borrow book X – or for recording a fact – "book X is now borrowed", or "the book has been returned", or for supporting a coordination indirectly – e.g. a reminder to return the book. We can use a web service for that, but strictly speaking, this web service is not a loan service: the loan is a transaction between the library and the client. The web service only supports certain parts of the messaging and data processing.

Now what about a much simpler example such as a currency conversion quotation? In this case, it looks as if service and web service coincide. It is true that the service

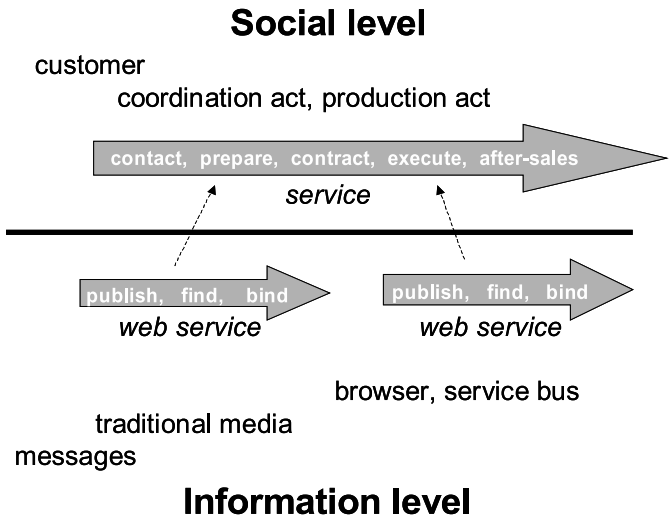

Figure 1: Social level versus information level of human action.

offered here (a production act) is highly automated. Nevertheless, it is not the web service that offers the conversion quotation service. A simple test question is: who may be held responsible by the client when a mistake is made, e.g. because of an out-dated conversion rate? Not the web service, but its provider. And what if there is no provider, if it is just a program hanging somewhere on the Internet? In that case, anybody can use it for what it is worth, but this use cannot be characterized as service provision.

Confusion between service and web service arises when the service is equated with agreement about a (future) service. For example, consider a train ticket, or a theater ticket. From a business point of view, the ticket is only instrumental: the real service offered is the travel or the theater show. The ticket represents a right and a commitment of the service provider. Information technology can be used for the customer to request such a ticket, and sometimes for the service provider to deliver the ticket in digital form. It does not provide the service as such.

Figure 2 renders a DEMO ATD (Actor Transaction Diagram) of a library (taken from [Di06]). One of the production acts offered by the library is the loan creation: this is done by the role “Loan Creator”. There are two transactions between Member and Loan Creator: ”loan start” and ”book return”. The former is aimed at producing the fact ”Loan L has been started”, and the latter at producing the fact ”book copy C has been returned”. Each transaction consists of a workflow loop with a initating phase (request, promise) and a completion phase (state, accept), and between these

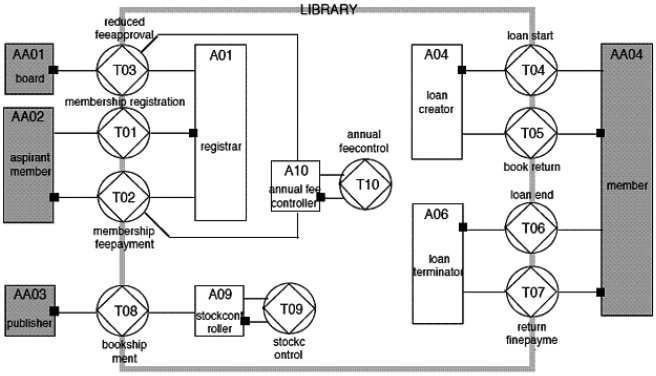

Figure 2: DEMO example library, extracted from [Di06].

two coordination acts, a production act is performed – the starting of the loan, or the return of the book. It is important to realize that these are essential actions that will always occur in the library, irrespective of how they are supported at the information level (by web services, or loan machines, etc).

2.2 E3-value

Another theoretical framework that is considered to assist in clarifying the distinction between web-services and services constitutes e3-value, which entails another paradigm for modeling and analyzing business transactions in the abstract.
e3-value [GAV00] is a modeling approach that is originally aimed at supporting the explorations of new business networks. For these explorations, process details are not relevant. What is important is whether a collaboration can be set up that provides value to all participants. Recently, e3-value is also applied for other purposes, such as business/IT alignment. We briefly introduce the basic concepts. An actor is an economically independent entity. An actor is often, but not necessarily, a legal entity. Examples include enterprises, brokers and end-consumers. A value object is something that is of economic value for at least one actor. Examples: cars, Internet access, stream of music. A value transfer represents one or more potential trades of value objects, whereas a value transaction is a reciprocal combination of value trans-

fers. Value transfers are modeled as lines between value ports (incoming, outgoing triangles), whereas value ports can be bundled in value interfaces (ellipses).

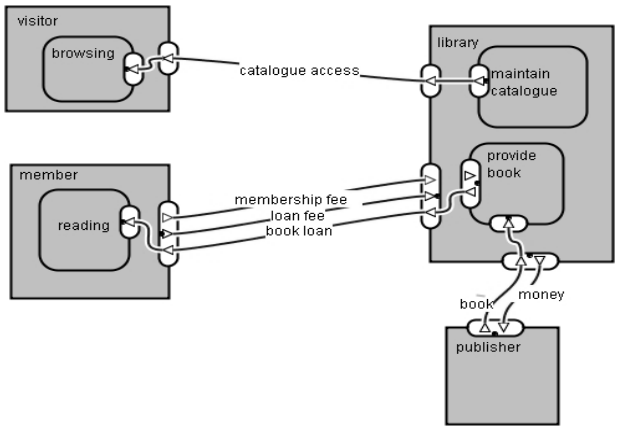

Figure 3: e3 value example library.

In [We06] the notion of value object is analyzed in more detail. Typically, it is a combination of an access right to some resource and a transformation enabled by this resource. For example, a flight as value object is the combination of a right to a seat in the named airplane and being transported to the destination. The analysis of the value object is helpful in identifying which coordination acts and production acts are needed to transfer the value.

Figure 3 presents a possible value model of a library. The library offers a loan service to members, but it also allows visitors (members or not) to use the book catalogue for browsing. As these services are independent, they are modeled as separate value interfaces. Within the value interface for ”book loan”, we see value ports for receiving the membership fee, for receiving the loan fee per book, and for providing the loan book.

Although both e3-value models and DEMO ATD’s aim at an abstract enterprise level, there are interesting differences. DEMO abstracts from information flow, and therefore does typically not include informational actions such as catalogue browsing. e3-value does consider all value objects, including informational ones. However, it abstracts from process aspects: for example, it typically does not make a difference between loan start and loan end – it just models ”loan” as a value object, and that this has a start and an end because of the semantics of ”loan” has to be considered when

going from value level to process level. In the following section, we will propose a conceptual architecture that links DEMO and e3-value. However, a full integration of the two methods is beyond the scope of this paper.

2.3 The conceptual architecture of services

The WSA [W304] uses the terms service and web service interchangably, although it does distinguish between a web service and its implementation. In this view, the web service is not the concrete piece of software, but the functionality provided by this software. Using Dietz's distinction axiom, the service is something offered at the performa or business level, whereas the web service functions at the informa level. Using e3-value, the provision of value can be made more precise. An important concept added by e3-value is the value transaction that includes the value transfer plus its exchange action (typically the payment). The notion of value transaction is important because there are often dependencies between the two value transfers (e.g. delivery only after payment). The term "service" is not used in e3-value, but it can be defined as the provision of a value object. Note that the value object is a type that can have multiple instances ("concrete services", in Preist's terms).
Figure 4 provides a metamodel relating the main concepts of a pragmatic service architecture that may be perceived as the first step in a unification of the Language/Action Perspective-, value modeling-, and the web service domain. A value transaction (e3-value) consists of at least two value transfers, here typified as a DEMO transaction that encompasses 3 phases. As such, a transaction realizes a certain service, that is, the provision of a value object by a provider to a requester.
Web services are organized in three groups: infrastructure web services, informational and domain web services. Infrastructure web services provide support facilities to domain services, e.g., persistency and lifecycle management, and may be packaged in middleware technologies such as the enterprise service bus. They function at the forma (documentary) level, whereas domain services in the service architecture operate at the informa level. Informational web services may automate information acts, whereas domain services may automate coordination acts or production acts. Hence, the latter may facilitate communication between a service requester and service provider to negotiate or comply to commitments using coordination web services, or, they may implement the actual value creation during the execution phase. Several web services for coordination can be distinguished, including web services supporting matchmaking and negotiation between the pre-transaction phase, and monitoring during the execution and post-transaction phase.

3 A pragmatic approach to service matching

It is a basic tenet of this paper that a web service is not the service, but an instrument at the informa level to support certain phases in the service lifecycle. The exact liaison between service (value transfer) and its lifecycle (process) is a topic of current

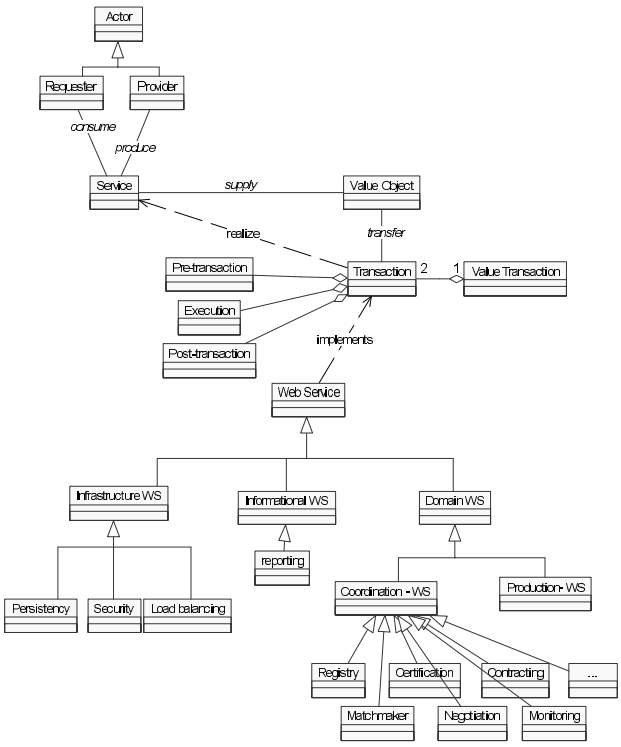

Figure 4: A unified metamodel defining and linking Services and Web Services.

research [We06] that is not in the scope of this paper.

Service discovery not only includes the use of semantic matching techniques (e.g., DAML-S matching), but it also has pragmatic aspects whose importance become apparent when we consider web services to support value transactions. Let us first explain what we mean by a pragmatic approach. Where semantics deals with the meanings of data, pragmatics deals with their context of usage [Si02]. Table 1 is an attempt to highlight the differences in focus between a Semantic Web approach and a Pragmatic Web approach.

The Semantic Web aims at improving information exchange. Its central object is the web document. Currently, many documents are only accessible to humans. The Semantic Web dream (Berners-Lee's words) is that agents can access and analyze these documents as well. To achieve that, documents must be enriched with semantic descriptions. These are typically based on ontologies, that are founded in Description

Logic and Knowledge Representation. In contrast, the Pragmatic Web aims at improving human collaboration, and its central object therefore is the (social) action. Currently, social actions are performed by humans. The Pragmatic Web challenge is that agents can effectively support humans in performing their social actions. To achieve that, agents must be able to enter meaningful conversations and get at agreements. These conversations can also function of hiding the "excess" semantics of data that might otherwise be revealed [Si02]. Ontologies do have a role here as well, but in principle, the agreements are grounded in the communities and their (mostly implicit) norms and trust relationships. Pragmatic Web solutions may come, among others, from Communication Theory and the field of Multi-Agent Systems.
Web services play an important role in both perspectives [Si02], but the Semantic Web efforts are mainly directed at finding web services, whereas a pragmatic approach relativizes this effort and is especially interested in service composition. Instead of adopting a light workflow-like language, like done by standards such as BPEL and OWL-S, the Pragmatic Web engineers processes as collaborations between "intelligent" services within a shared and trusted context. This implies that service composition should go far beyond connecting port types of (web-)services, and should involve issues such as reaching agreement about business protocols, security, transactionality, while assuming that semantic interoperability issues are handled by the community or vertical domain, e.g., using reference models such as RosettaNet. Once services have been composed, the Pragmatic Web applies dynamic service adaptation in context to modify functional and non-functional characteristics of services given external or internal stimuli. In [WHfc], we have explored some initial ideas regarding dynamic service adaptation.
In addition, and perhaps more importantly, the Pragmatic Web treats processes as value-exchanges, providing existential meaning and context to services. In this emerging paradigm, web-services constitute the main fabric to assemble processes that implement services and take care of value transactions between service requesters and providers.

Semantic Web	*Pragmatic Web*
information exchange	collaboration
document	action
agents can analyze documents	agents can perform delegated actions
semantic descriptions	conversations and agreements
ontologies	communities
Description Logic, Knowledge Representation	Communication Theory, Multi-Agent Systems
finding services	composing, adapting services

Table 1: Semantic Web vs. Pragmatic Web: distinctive and complementary

3.1 Service discovery

The conceptual architecture of Preist contains three models, the first being the discovery model. The model assumes that both requestor agent and provider agent have descriptions of their requirements and offers respectively. These descriptions can be matched. It is noted that a match does not guarantee that the service can be actually delivered – the item may be out of stock temporarily. We will come back on the availability check in the next subsection.
According to [LO05], the discovery process takes two steps. First, using abstract service descriptions written in Description Logic [LH03] that only considers the desired result, and one that takes into account actual input, where transaction logic can be used to simulate an actual service execution. The latter is not based on actual availability checking, but is based only on the abstract service specification. The actual availability check is done during service contracting. The two steps proposed in [Fe05] are similar, but they do not commit themselves to specific logics. In both approaches, the first match is done on what they call an ontological level, that is, in terms of the service description rather than the web service (WSDL) description, and this is a significant improvement. Nevertheless, in our view both approaches still link web services and services too closely. The first step is called web service discovery, but what is done is matching of service descriptions. So first of all, the client is trying to discover services, not web services. One possible way is to look into each web service whether the required service is offered there, but this can also be done via other ways (e.g. the service provider may have advertised his services somewhere on the Internet or outside). Once a service is identified, the user can look for its web service, if any. In the second, or now third step, a more refined match can be made. However, it is not evident that this should be based on the choreography of the web service. The input requirements can also be matched at service level first, including a comparison between competitive providers. And only after this has been done successfully the web service requirements are considered. Below, we will propose a flexible way of setting up web services so that the restrictions of the choreography are minimal.
The identification of an abstract service should be based on the value object it offers. Several semantic techniques can be used to improve the matching process in terms of recall and precision [MM03]. From a value perspective, it is important to be aware of the ”means” relationship between resource and value transformation. A client typically is interested in the value transformation, and the provider in providing a resource. For example: the client wants to get from Schiphol Airport to Amsterdam. A resource offered by Rent-a-Car can be a loan car. ”car” and ”getting to Amsterdam” do not match directly, only if we recognize that the former can be a means to achieve the latter. Some providers may address both the resource and the value transformation in their service description, but since this will not be always the case, we recommend the use of a general means/end relationship acquired by learning.

Secondary values are the features (of value to the customer) that within a certain industry distinguish one service provider from another. For example, there are many companies that can offer flights; some of them provide the secondary value cheap, other the secondary value comfortable, etc. For a good matching of service needs and services offered, the secondary values are a major means of selecting the most suitable service for a customer.

3.2 Reducing uncertainty

From a pragmatic point of view, the main objective of the pre-transaction phase is to reduce uncertainty about the value transaction (that is, to realize the value transaction and mitigate the risks involved such as items being out of stock). Service discovery should not only consider static published service descriptions. The following characteristics of e-business transactions were identified in [AW05]

- **Uncertainty about Functionality**. Providers are dynamic (volatile) – while initially advertising their functionality in UDDI-type registry, they might change as a whole or some of its characteristics
- **Uncertainty about Providers**. Providers are autonomous entities, sometimes exhibiting opportunistic behavior
- **Uncertainty about Technical Infrastructure**. The transactions' technical infrastructure (media, protocols, etc.) might be unreliable
- **Uncertainty about Contractual Terms**. Rather than being focused on execution speed, e-business transactions use schedules and timeouts (specified duration); being contract-governed execution, duration is predefined.
- **Uncertainty about Capacity**. Participants have functional and capacity restrictions on their operations. Functional reflect internal business capabilities of the participants that are hard to change, capacity addresses participants' characteristics at the execution time.

In order to cope with these business uncertainties, [AW05] proposed an advanced locking mechanism. The pre-transaction might be split into two sub-phases: prepare and locking. Technically, e-business locking is an asynchronous message exchange between prospective participants and a change of the Provider's state in case lock is applied successfully. While not being explicitly addressed, e-business transaction execution is enclosed with support/service phases addressing reliability and correctness issues.
We slightly adapt [AW05] to the context of services and web services and suggest a prepare and a reserve phase with the following characterization:

Prepare phase – this includes functionality verification of prospective providers. It precedes both reservation and execution. We assume this phase's activities verify functionality (that is, the value object) of prospective providers and do not impose any definite reservation, neither they impose an obligation on any party. In case of completion or execution failure the preparation does not require any compensation. Because the provider's profile, initially published in a registry might reflect actual functionality incorrectly (being outdated) or incompletely (containing insufficient information to invoke provider's functionality), the requesting party might want to check this information. An example is checking with a supplier whether a certain component can be ordered.
While functionality is quite stable, provided capacity (value object instances) may vary in a rather unpredictable manner due to resources' utilization by other parties. Exact capacity value (or, rather, available capacity) is correct only at the reserve phase, however, any estimate performed before invocation also contributes to the efficiency of execution scenario because it allows excluding (potentially) unavailable participants from the scenario right from the start. Following up on the example, the capacity check asks whether the specified chip will be in stock at some date in the future.

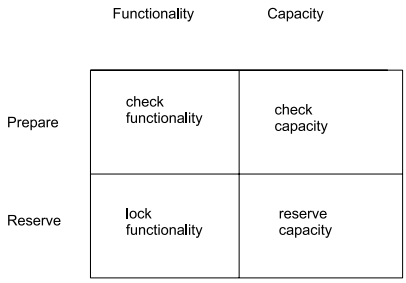

Figure 5: Functionality vs. capacity and prepare vs. reserve

Reserve Phase – sometimes it is useful to reserve capacity before actually committing to buy it. The reserve phase follows the prepare phase and typically it assumes the existence of a contract or similar agreement specifying conditions of cancelations and non-performance, but this contract can also be more general and implicit.
Checking applies either to the functionality and the capacity. The same is true for the reservation, which leads to an orthogonal architecture (figure 5). This orthogonal architecture provides the following benefits:

- it allows transitivity of properties and functionality from participant (Provider) to capacity (resources) it controls, thus optimizing speed of reservation (no need to provide additional information for every operation);

- it minimizes cost and impact of compensation. Application of participant/ capacity reservations could be relatively extended in time, allowing reservation costs to be minimized. For example, a check can be performed in January, a provider reservation in March (for the rest of the year), a capacity reservation in June, when the production planning is finalized, while the actual execution is only performed in October.

  When the check in January fails, another supplier can be looked for. Similarly, when the provider reservation fails in March, another short-listed supplier can be chosen. In this way, the risks are minimized against minimal costs.

Both prepare and reserve phase precede transaction execution, but their impact is quite different. While check verifies functionality and requests additional information, reserve is applied upon known functionality; the check request is based on advertised functionality, while reserve is based on confirmed (verified) functionality; in the case of reserving, compensations might follow for cancelations, while check is a request for information with no compensations defined or needed. The provider is considered to be a prospective one before reserve application and actual after.

4 Conclusion

Service-Oriented Architectures provide major advantages for today's enterprise information systems by presenting the interfaces that loosely coupled connections require. Web services seem to become the preferred implementation technology for realizing the SOA promise of service sharing and interoperability. In this paper, we have scrutinized the difference between service and web service. Two complementing modeling techniques were presented that support the modeling of services at an abstract, technology-independent level. Several pragmatic aspects of service discovery have been discussed, including the use of uncertainty-reducing techniques during web-service discovery.
The grand challenge for the research community for the coming years is to develop and evaluate well-founded and pragmatic design methodologies for web services. The two theoretical frameworks presented in this paper have proven to be useful in enterprise modeling. A topic for future research is to integrate these frameworks and connect them with a Service-Oriented Architecture, so that web service design can be performed at business level supported by automatic or semi-automatic generation of web services.

References

[AW05] S. Artyshchev, H. Weigand. Interoperable transactions for E-Business. In *Proc. 16th IFAC World Congress*, Prague, 2005.

[Di06] J. Dietz. *Enterprise Ontology – Theory and Methodology*. Springer, Berlin, 2006.

[Fe05] D. Fensel et al. WWW or What is Wrong with Web Service Discovery. In *W3C Workshop on Frameworks for Semantics in Web Services*, Innsbruck, Austria, 2005.

[GAV00] J. Gordijn, H. Akkermans, J. van Vliet. Business Modeling is not Process Modeling. In *Conceptual Modeling for E-Business and the Web*, Berlin, LNCS 1921, 2000. Springer.

[LH03] L. Li, I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, 2003.

[LO05] R. Lara, D. Olmedilla. Discovery and Contracting of Semantic Web Services. In *W3C Workshop on Frameworks for Semantic in Web Services*, Innsbruck, Austria, 2005.

[MM03] D. Mandell, S. McIlraith. Automating Web Service Discovery, Customization, and Semantic Translation with a Semantic Discovery Service. In *WWW (Posters)*, 2003.

[Pa05] M. Papazoglou. Web Services Technologies and Standards. *ACM Computing Surveys*, to appear, 2005.

[Pr04] C. Preist. A Conceptual Architecture for Semantic Web Services. In Frank van Harmelen Sheila A. McIlraith, Dimitris Plexousakis (ed.), *The Semantic Web ISWC 2004: Third International Semantic Web Conference*, Berlin, LNCS 3298, 2004. Springer.

[Si02] M.P. Singh. The Pragmatic Web: Preliminary Thoughts. In *Proc. of the NSF-OntoWeb Workshop on Database and Information Systems Research for Semantic Web and Enterprises*, April 2002, pp. 82-90.

[W304] W3C. *Web Services Architecture W3C Working Group*. http:www.w3.orgTRws-arch, 2004.

[WHfc] H. Weigand, W.J. van den Heuvel The challenge of self-adaptive systems for E-commerce. In *Group Decision and Negotiation*, forth-coming

[We06] H. Weigand et al. On the Notion of Value Object. In *Proc. CAISE '06*, Berlin, 2006. Springer.