# Value-based Service Design Based On A General Service Architecture

Hans Weigand[1], Paul Johannesson[2], Birger Andersson[2], Maria Bergholtz[2], Ananda Edirisuriya[2], Tharaka Ilayperuma[2] , Jelena Zdravkovic[2]

[1] Tilburg University, P.O.Box 90153,
5000 LE Tilburg, The Netherlands
H.Weigand@uvt.nl,
[2] Royal Institute of Technology
Department of Computer and Systems Sciences, Sweden
pajo,ba,maria,si-ana,si-tsi,jelenaz@dsv.su.se

Service-oriented architectures are the upcoming business standard for realizing enterprise information systems, thus creating a need for analysis and design methods that are truly service-oriented. Most research on this topic so far takes a software engineering perspective. For a proper alignment between the business and the IT, a service perspective at the business level is needed as well. In this paper, definitions of "service" are analyzed, resulting in a set of models for the service concept. On the basis of this service architecture, a service design method is proposed and applied to a case from the literature. The design method capitalizes on existing value modeling approaches.

## 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 [WS04] seem to become the preferred implementation technology for realizing the SOA promise of service sharing and interoperability.

Papazoglou and Van den Heuvel [PH06] provide an overview of methods and techniques used in service-oriented development and design. In their view, (web) service design and development is about identifying the right services, organizing them in a manageable hierarchy of composite services and choreographing them together for supporting a business process. A *business service* (in this context, a service implementing a business process) can be composed of finer-grained services, which in turn are being supported by infrastructure services. Following previous work on SOAD [Zi04], they distinguish between top-down, bottom-up and meet-in-the-middle approaches and discuss major principles of service design such as low coupling and high cohesion. With respect to service granularity, it is recommended to build higher-level, coarse-grained interfaces that implement a complete business process and that are composed of fine-grained services designed for reusability. Although the paper provides useful criteria, it considers service design mainly as a software engineering problem, which in our view is not sufficient. As stated in

[NL07], "the current trend toward a service-oriented enterprise necessitates a formal characterization of business architecture that reflects service-oriented business thinking". The starting point for design should be the business level at which services can be identified that provide value to customers and can be offered in an economically viable way.

The objective of this paper is to develop a method for the service identification task that is value-based (rather than software-based) and comprehensive (covering the various different aspects of services). Section 2 sets the stage by a short overview of some prominent business modeling approaches and an analysis of definitions of "service". In section 3, we formalize the service concept in a couple of models. Section 4 applies this service architecture to service design using an example from the SOA literature, so that the results can be compared. Section 5 contains the conclusions and directions for future research.

## 2. Business Modeling and Service definitions

As we want to relate notions of business (or value) models with notions of service models we begin this section with an overview of business modeling. There exists a number of approaches, languages, and ontologies for business modeling in literature, e.g., [Go00], [Di05], [Us95] and [Mc82]. For the purpose of this paper we make use of one comprehensive and well established business model ontology, the e3value [Go00], that is widely used for business modeling in e-commerce. In [An06] the e3value and the REA ontologies [Mc82] were compared (together with a third business ontology - the BMO [Os04]) in order to establish a common reference business ontology. One result of that comparison was a set of mappings between e3value and REA indicating strong similarities between the concepts of the two.

### 2.1 Business Modeling

The *Resource-Event-Agent (REA) ontology* was formulated originally in [Mc82] and has been developed further, e.g. in [Ge99] and [UM03]. Its conceptual origins can be traced back to traditional business accounting. REA was originally intended as a basis for accounting information systems and focused on representing increases and decreases of value in an organization. REA has been extended to form a foundation for enterprise information systems architectures [Hr06], and it has also been applied to e-commerce frameworks [UM03].

The core concepts in the REA ontology are Resource, Event, and Agent. The intuition behind the ontology is that every business transaction can be described as an event where two actors exchange resources. To acquire a resource an agent has to give up some other resource. For example, in a goods purchase a buying agent has to give up money in order to receive some goods. The amount of money available to the agent is decreased, while the amount of goods is increased. Conceptually, two events are taking place: one where the amount of money is decreased and another where the amount of goods is increased. This combination of events is called a duality and is an expression of economic reciprocity - an event increasing some resource is always accompanied by an event decreasing another resource. A corresponding change of availability of resources takes place at the seller’s side. Here the amount of money is increased while the amount of goods is decreased.

There are two types of events: exchanges and conversions [Hr06]. An exchange occurs when an agent receives economic resources from another agent and gives resources back to that agent. A conversion occurs when an agent consumes resources to produce other resources. Events often occur as consequences of existing obligations of an actor; in other words, events fulfill the commitments of actors.

The *e3value value ontology* [Go00] aims at identifying exchanges of resources between actors in a business case and it also supports profitability analyses of business cases. The ontology was designed to contain a minimal set of concepts and relations to make it easy to grasp for its intended users. e3value includes a graphical notation for business models. The basic concepts in e3value are actors, resources, value ports, value interfaces, value activities and value transfers (see Fig. 1).

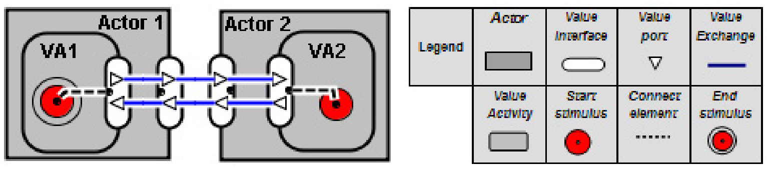

**Fig. 1** Basic e3value concepts

An *actor* is an economically independent entity. An actor is often, but not necessarily, a legal entity, such as an enterprise or end-consumer or even a software agent. A set of actors can be grouped into a market segment. A resource (also called *value object*) is something that is of economic value for at least one actor, e.g., a car, Internet access, or a stream of music. A *value port* is used by an actor to provide or receive resources to or from other actors. A value port has a direction: in (e.g., receive goods) or out (e.g., make a payment), indicating whether a resource flows in to or out from the actor. A *value interface* consists of in and out ports that belong to the same actor. Value interfaces are used to model economic reciprocity and bundling. A *value exchange* represents one or more potential trades of resources between these value ports. A *value activity* is an operation that can be carried out in an economically profitable way for at least one actor.

Both the e3value and the REA ontologies include concepts on the operational level as well as the knowledge level [Fo97], where the operational level models concrete, tangible individuals in a domain, while the knowledge level models information structures that characterize categories of individuals at the operational level. In REA almost all classes on the operational level have a corresponding class on the knowledge level, which is generally not the case for e3value. The REA ontology distinguishes between event type (abstract transfer of categories of resources) and event (actual transfer of tangible concrete resources), both of which correspond to e3value’s value transfer. The economic reciprocity between two REA events is modeled through the duality relationship, while the same reciprocity between value transfers in e3value is accomplished by bundling the value transfers through value interfaces.

## 2.2 Service Definitions

In this section we continue the discussion from section 1 regarding the term service, its definitions, and its use in different contexts, in particular business and software. We have found many definitions in the literature, the most important ones being:

**W3C:** "A service is an abstract resource that represents a capability of performing tasks that represents a coherent functionality from the point of view of provider entities and requester entities. To be used, a service must be realized by a concrete provider agent. A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL)." [WS04].
**OASIS:** "A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description." [OA06].
**WSMO:** "A service in contrast is the actual value provided by this invocation. Thereby a Web service might provide different services, such as for example Amazon can be used for acquiring books as well as to find out an ISBN number of a book. A WSMO Web service is a computational entity which is able (by invocation) to achieve a users goal." [WS05].
**Preist:** "A service is the provision of something of value, in the context of some domain of application, by one party to another. We need to distinguish between a particular provision of value from the general capability to provide. We refer to the former as a concrete service, and the latter as an abstract service. Hence an abstract service is defined as the capacity to perform something of value, in the context of some domain of application. An agreed service is an abstract service agreed between two parties." [Pr04].
**UN:** "Services are heterogeneous outputs produced to order and typically consist of changes in the conditions of the consuming units realized by the activities of producers at the demand of the consumers." [UN08].

In the above list, a common view of a service among the sources is that a service is an abstraction of transformation or communication work, and once started it will achieve some user's goal(s). However, the exact way of defining the service depends on the perspective of a particular source. For example, Preist, UN, and OASIS focus on a business service perspective, while W3C and WSMO have a web (or software) service perspective.

In the business service perspective, a service is defined by relating it to the value it creates for the users, or seeing it as a way of giving them an access to business processes. Analogous to a software service a business service hides the actual implementation details of a business process. Defining a service as an abstract knowledge level concept gives providers the flexibility of implementing it in different ways in order to offer it to service users. Preist describes this as distinguishing between a particular provision of service from the general ability to provide it. A business service, as well as the common concepts customer (or user) and provider, are defined at the type level [Pr04, WS04].

When a service is offered by a particular provider to a particular customer the actual number and types of resources used in the implementation of the service and the way of offering it can be different. In other words, one instance of a service type (a realization) may differ from another [Wi08, UN08, Pr04]. For example, in a hair cutting service, hair cutting could be done in different ways by using different resources.

The service realization is done with a goal of meeting certain demands of service users [UN08, Wi08]. For example, a particular user may want to get a hair cut fast forcing the provider to use a cutting machine instead of scissors and skip regular tasks like shampooing, etc. A service realization and provisioning is done in compliance with the business policies and constraints of the provider organization [OA06].

A *web service* is defined in two ways. In [WS04], a web service is defined as a kind of a business service involving only software applications. Hence, it seems that in W3C, the business service is more general than the web service. In [WS05], a web service can provide more than one services. For example, Amazon can be used for acquiring books as well as to find out an ISBN number of a book and for them these are distinct services provisioned by a single web service.

The [WS04] and [WS05] definitions unveil an important aspect of a service in general - it's compositeness. That is, a service may consist of more than one service that will fully realize the first service when it is invoked. This means that regardless of whether a service is either a general business service or a web service, it could be a container of several other services that fully realizes the goals of that service. Below we summarize the main points from the above analysis of service definitions.

- A service is an abstract resource with a well-defined interface.
- A service is always attached to a provider agent and a customer agent.
- A provider may realize a service by performing various tasks that create some value to service users.
- The abstractness of a service makes it possible to realize it in different ways by using different resource required by the service itself, and also meeting the goals and needs of both the provider and the customer.
- The realization of a service may be carried out according to the policies and constraints of both the provider and the customer.
- A service can be composite meaning that it requires (depends on) one or several other services in order to fulfill its goals.

In the following section, we use these different characteristics identified by analyzing the definitions above, to propose models covering various aspects such as delivery, realization, provisioning, composition, etc., of a service. If we want to talk specifically about services realized in SOA software, we use the term "web service".

## 3. Service models

In this section, we introduce a conceptual service architecture that bridges the business and software level. Although it is often claimed that service design should be guided by a business perspective, the current literature is not clear on how this should be done. Our starting point is the W3C web service architecture (WSA), which although focusing on web services develops a framework which in fact is much more general. In addition, we use basic concepts of the REA business ontology described above. As in WSA, we use the term "architecture", for what could be called an ontology or conceptual model as well.

Like WSA, we will use several models in order to highlight different essential aspects of services. The two basic ones are the service realization and service delivery models that address the *what* and *who* questions. In addition, the service provisioning model takes a provider's perspective (*how*). Not included in the WSA but quite important in SOA, is the service composition model. Finally, we describe a service policy model that captures the context of services.

## 3.1 Service realization

According to WSA, a web service is an abstract notion that must be realized by a concrete agent. The same service can be realized by different agents and different pieces of software. Following this conceptualization, we define a service at the knowledge level and distinguish it from a service realization being an event happening at some specific time. Following REA, the essence of the service is a conversion of some resource; we call the resource type on which the service works the subject of the service. The goal of the service is some state of the subject; in an abstract way, the service could be identified with the state change it engenders, but in business practice, this state change is only partially defined. Conversion of a resource also covers the special cases of creation or destruction of a resource. The realization of the service is typically done by means of actions not worked out further.

*Example*: A flight service transports persons (subject) to a destination (state condition), using air craft (resource type) for consumers and business people (stakeholder segments). A flight service realization transports Mr. Jones from Amsterdam to Stockholm; actions include boarding, take-off, flight, and breakfast.

Traditionally, in business economics a difference is made between goods and services. For instance, according to [BG04] 'services' in the business science community are "business activities that result in intangible outcomes or benefits". However, from the point of view of the buyer the purchase of a resource (a car, a loaf of bread) engenders the availability of that resource, and hence it is possible to treat this kind of transaction under the same service umbrella. From an (admittedly biased) service perspective, the broader definition of service is to be preferred.

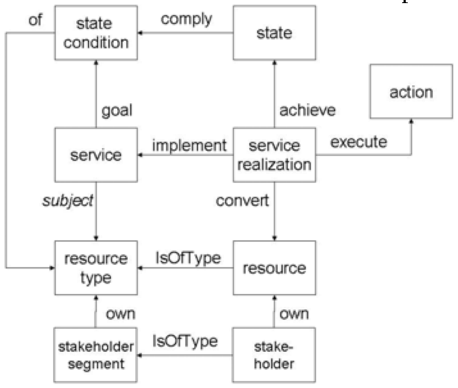

**Fig. 2.** Service Realization Model. Italicized labels stand for indirect relationships

### 3.2 Service delivery

According to WSA, the purpose of a web service is to provide some functionality on behalf of its owner entity (provider) to another entity (requester). Whereas the service realization model focuses on changes brought about by the service in the *object world*, the service delivery model positions the service in the *social world*: a service means someone serving someone else (cf. [Di06]). A service delivery is the event in which a service realization is delivered by provider to requester. The latter are actors, which can be persons, organizations or pieces of software performing a function on behalf of a human actor. The service delivery being an event is an abstraction, as typically the delivery is realized in several steps, such as contacting, negotiation and completion (more on this below in 3.4). The provider often coincides with the one who realizes the service (agent, in WSA), but the realization may have been delegated to another actor (including a piece of software).

The service delivery model extends WSA with the notion of transaction. Instead of providers simply offering services, it is assumed that they offer services typically as part of a transaction (an atomic exchange of multiple services that accounts for reciprocity and service bundling), as in the e3-value ontology.

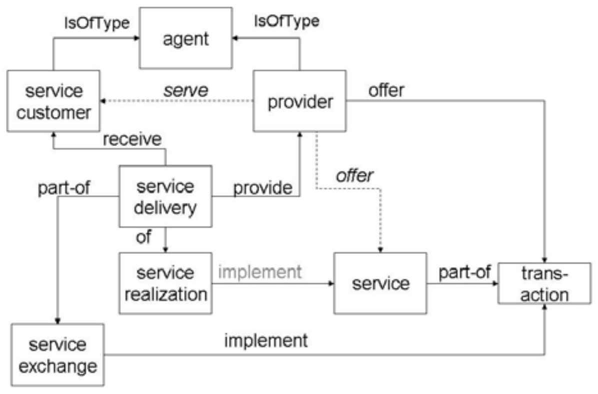

**Fig. 3.** Service Delivery Model. Grey labels stand for earlier-defined relationships, and the dotted lines for derived relationships

*Example*: A flight service transaction offered by KLM includes a flight service and a payment. KLM may deliver a flight service realization such as mentioned above to Mr. Jones. The service exchange includes this realization and the payment transfer of €190. When there are 80 passengers on board, there are 80 service deliveries and 80 realizations of the same service (ignoring flight class distinctions).

Identifying a service transaction with a value transaction, a service with a value object and a service delivery with a value exchange, we can build a bridge between the service architecture and the $e^3$*-value* ontology. This makes it possible to use a value model as starting point for the design. Internal value object exchanges (between value activities of one actor) are treated interpreted in the same way.

### 3.3 Service provisioning

Service provisioning is the third model that provides the back-office view of the service: how the provider is able to deliver the service, what are the resources used, in contrast to the resources acted upon. In some sense, the service provides access to these resources or capabilities (cf. OASIS definition). They are owned by the service provider or obtained by him in some way, and exploited in the service realization.

*Example*: The flight service offered by KLM makes use of aircrafts; a concrete resource may be a particular Boeing 737. Although the flight may be executed by a third party, the resources actually used coming from several organizational actors, KLM is responsible for the realization. Other resources include aircraft personnel.

Although the service provisioning model usually does not need to be published, it is relevant to service design. One can only offer services that one is able to provision. The service provisioning model prompts the designer to indicate what resources are needed and to reflect on where they come from. From a business engineering perspective, the reverse route makes equal sense: starting from the resources of the organization, how can these resources be used to serve customers on an economically viable way and in this way can be developed further?

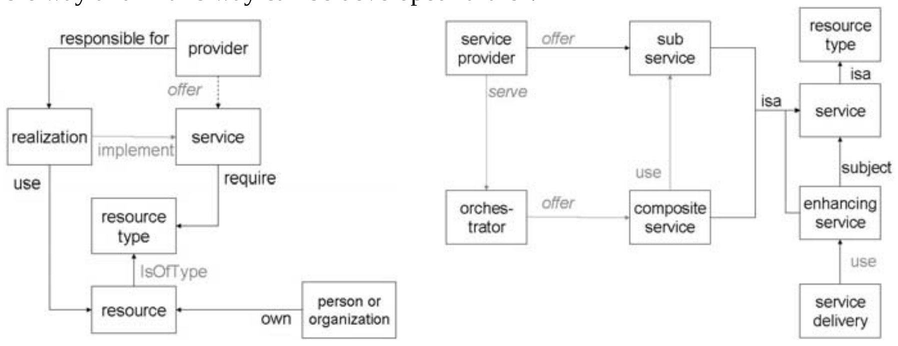

**Fig. 4.** Service Provisioning and Service Composition Model

### 3.4 Service composition

Service composition is a key issue in SOA, but not dealt with in WSA, presumingly because of its focus on the interaction between user and web service. On the basis of the REA ontology underlying the previous service models, we can distinguish between two kinds of composition, deriving from two kinds of relationships between services. On the one hand, services can *use* other services as resources. This case builds on the Service Provisioning Model. On the other hand, services can act upon (*convert*) other services.

*Example* A typical example of a service using other services is the often recited travel agency service that uses hotel services, flight services and others. Hence the travel agency service is a composite service.

A service can also be a resource acted upon by another service called an enhancing service in this case. Typical examples of enhancing services are the services that provide access to the service in question (the core service), for example a contacting service, a reservation or negotiation service (cf. [Go06]). What these services have in common is that they assume the core service to be in place independently; otherwise, they could not claim to inform about this service, or reserve it. Note that they are not about the abstract service, but about a service realization by a specific provider. The enhancing services are closely related to the service delivery; therefore an alternative characterization is that they are resources "used by" a service delivery event.

*Example* Enhancing services for the flight service are flight reservation, flight information and check-in – all of these may be delegated partly or completely to web services.

### 3.5 Service policy model

The service policy model accounts for the context that constrains the realization and delivery of the service. Drawing on the previous service models, the policy can be modeled as a service itself that aims at making the service comply with certain rules. The policy is issued by an authority (viewing the policy as a service, the provider offers the policy realization to some authority that, in terms of WSA, establishes the policy). This may include the provider and customer. In WSA, policies "apply to" services; in our model, the policies apply indirectly to service providers as well.

*Example* To KLM as flight service provider, many policies apply to various services that it offers (the flight, the check-in etc). The realization of these policies is delivered by KLM to these authorities, the goal being policy compliant. KLM itself may also issue policies internally.

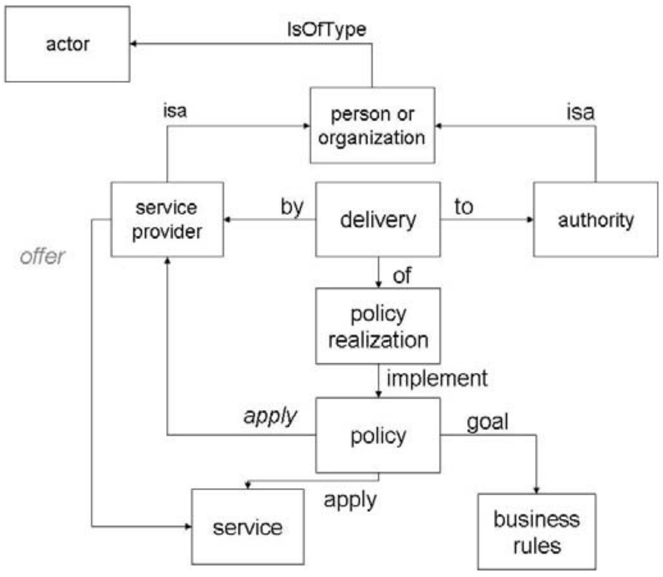

**Fig. 5.** Service Policy Model

## 4. Service design – a method proposal

The service models introduced in section 3 enable the designer to start designing or identifying services in the business domain, and from there design web services in the information system domain. As in [HJ07], we assume the designer does not need to start from scratch, e.g. that a value model has been developed first – the advantage is that this activity is already supported by established modeling techniques and methodologies. Moreover, this means that essential decisions, as for the choice between using internal or external services (in which strategic concerns related to trust and collaboration play a role), have been made already (represented in the positioning of the value activities). In section 4.1 we introduce a value-based service design method. In 4.2, this method is applied to an example from the research literature: the Automotive Work Order case described in the IBM paper on SOAD [Zi04].

### 4.1 A service design method

The service design method proposed here is based on the service models that we have developed and takes the following steps. Starting point is the value model. Step 6 marks the transition to web service design.

1. For each value object in the value model, introduce one service (called core service). This step refers to the part of the Service Realization Model that deals with what abstract core services exist. Then consider in which value transfers the value objects are exchanged and derive which service deliveries are made by which actors, as part of which transactions (Service Provisioning and Delivery Model).
2. If more than one value activity is distinguished within a certain actor, they give rise to core services as well. Furthermore, each value activity corresponds to an (internal) actor that provides this service to the actor in which the value activity is enclosed.
3. For each core service identified in 1 and 2, introduce the resources needed.
4. For each resource identified in 3, indicate the service that realizes this resource. If the service was not identified yet, introduce a new one. This step draws again on the Service Provisioning Model. The step is applied recursively till the scope of the model is reached.
5. For each service identified in 1, 2 or 4, introduce additional enhancing services. The enhancing services identified in this step are services that do not provide resources per se but are used to realize the earlier identified services by providing access to them (cf. Service Composition Model)
6. For each service identified in 1, 2, 4, or 5, try to delegate the service to a web service. This is only possible when the service is informational in nature. Core services can be informational, as in the case of digital products. In most cases, the enhancing services are suitable candidates for automation.

The design steps identify services, not their relationships. To glue the services together, we need an orchestration such as a BPEL[BP07] specification or business rules (see below). The capabilities of each service are not derived from the *$e^3$-value* model, but we hypothesize that a major part of e.g. a negotiation service interface can

be inherited from generic negotiation services (e.g. the Conversation for Action pattern [Di06,Go06], or some domain-dependent variants of it). Such an approach can be seen as a concrete implementation of the suggestion made by [PH06] that service design should draw on business standards whenever possible.
If enhancing services are automated, it is important to note that if service A uses service B, then the enhancing service for A *may* have to use the enhancing service of B. We call this the *analogy principle*.

7. Classify the web services using available generic web services.
8. Web services that use other web services are worked out as composite services. For each composite service, introduce policies that contain the business rules applying to the composition.
9. Consider the usefulness of combining small web services into larger ones (if they are strongly connected), or introducing intermediary web services (e.g. adapters). The design method suggests how to arrive at an initial web service model, but it does not remove the need to take design decisions anymore. An important factor at this point is the extent in which the organization wants to reuse its existing components or is willing to invest in service-oriented redesign.

### 4.3 Example: The Automotive Work Order

The Automotive Work Order domain describes the process by which an automotive maintenance company manages its customer operations. This example has been used to illustrate SOAD in [Zi04].

A *work order* represents an agreement between the auto service company and a customer to perform a set of scheduled or emergency maintenance activities such as a scheduled 50,000-mile service, a brake pad or tire replacement, or an oil change.

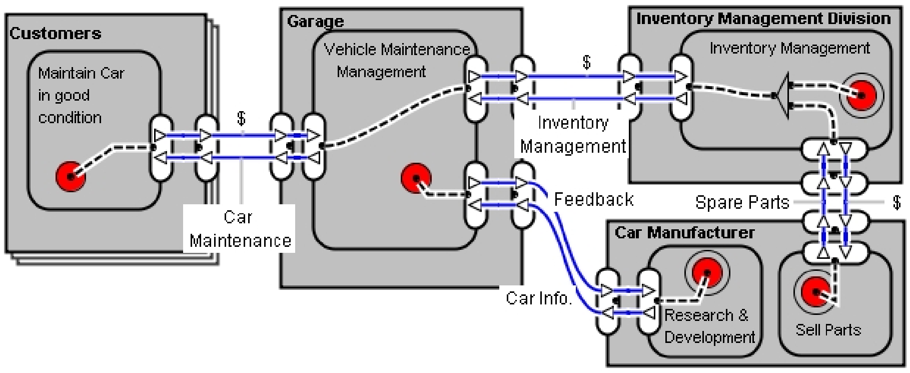

**Fig. 7.** Value Model for the Automative Work Order Case

The above $e^3$*value* model describes how the automotive maintenance company manages its customer operations. Mainly there are three actors in the model: The Customers, the Automotive Maintenance Company (later referred as the Garage + Inventory Management) and the Car Manufacturer. The scenario is that the Customers contact the Garage and schedule a time for the maintenance work. The garage makes

sure, through the inventory management, that the necessary parts needed for the repair is available before the work begins.
Starting from the $e^3$*value* model and following the methodology steps of the prior section, we arrive at the following.

1. For *each value transfer in the value model, introduce one core service*:
S01 Car maintenance (provided by G to Customer, subject car, part of commercial transaction)
S02 Inventory supply (provided by IM to G, subject part, part of internal transaction)
S03 Parts supply (provided by CM to IM, subject part/supply, part of commercial transaction)
More services will be identified if we include the reciprocal value transfer (payment) and the Car Manufacturer, but in line with the [Zi04], we delimit the design to the Vehicle Management point of view.

2. *Similar for value activities*.
Only one value activity exists for the actors considered, so step 2 does not apply.

3. *For each core service identified, introduce the resources needed*
Analyzing the service provisioning, the following resources can be identified:
R01 Resources: bay, mechanic, parts, supplies, plus competence
R02 Resources: parts, supplies, plus competence
R03 Resources: parts, supplies, plus competence

4. *If needed, introduce services that provide the resources*
These resources, except for competence, need to be obtained. Focusing on S01, the following additional services can be identified (for parts and supplies, we already have service S02):
S04 Work floor service (provided by G1 to G, subject bay, part of internal transaction)
S05 Human Resource service (provided by G2 to G, subject mechanic, part of internal transaction)
S06 Car drop service (provided by Car Owner to G, subject car, part of commercial transaction)

5. *For each service identified in 1, 2, or 4, introduce additional enhancing services*
The most relevant is Negotiation (the service that aims at establishing an agreement on a core service to be provided). Following the analogy principle, the Negotiation service for S01 will have a dependency on the Negotiation services of S02, S04-S06. As the Negotiation is part of the encompassing delivery service, we identify a Maintenance Delivery (composite) service as well.

6. *Identify possibilities of automation*.
7. *Classify the services using generic web services and reify the interface*.
8. *Work out composite services*.
9. *Rearrange web services if needed*

First of all, all the Negotiation services could be delegated to a web service. We need an IT Infrastructure service to realize these web services to the AMC. The Garage Negotiation service provides the interface to the Customer, and will be realized as a composite service as it calls the other services. The service that the negotiation service provides takes the form of a delivery of an agreed work order.

The provisioning of knowledge resources can also be automated, as they are informational in nature. For example, the Garage has knowledge about the possible maintenance tasks. As a result, we obtain the following web services:

W01 Car Maintenance (negotiation) Service (subject work order)
W02 Work floor (negotiation) Service (subject bay reservation)
W03 Human Labor (negotiation) Service (subject mechanic reservation)
W04 Inventory (negotiation) Service (subject part reservation)
W05 Garage Knowledge Service (subject maintenance knowledge) (other knowledge services omitted here)

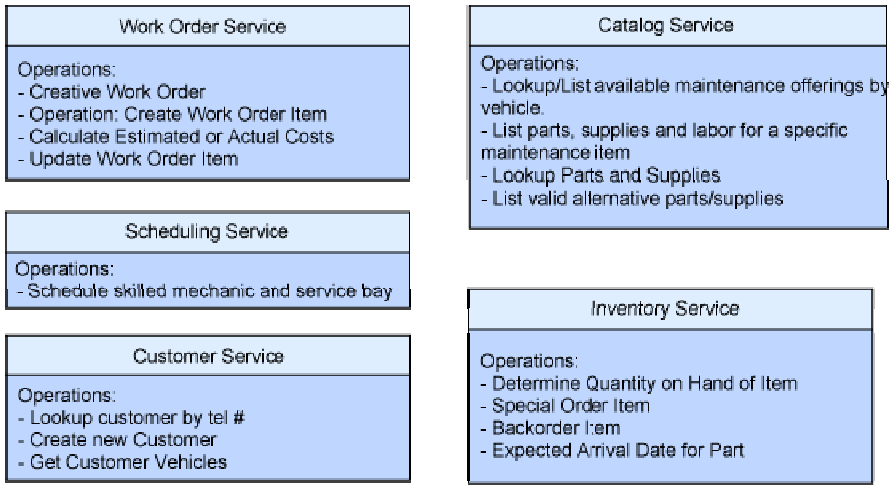
Work Order Service
Operations:
- Creative Work Order
- Operation: Create Work Order Item
- Calculate Estimated or Actual Costs
- Update Work Order Item
Catalog Service
Operations:
- Lookup/List available maintenance offerings by vehicle.
- List parts, supplies and labor for a specific maintenance item
- Lookup Parts and Supplies
- List valid alternative parts/supplies
Scheduling Service
Operations:
- Schedule skilled mechanic and service bay
Customer Service
Operations:
- Lookup customer by tel #
- Create new Customer
- Get Customer Vehicles
Inventory Service
Operations:
- Determine Quantity on Hand of Item
- Special Order Item
- Backorder Item
- Expected Arrival Date for Part

**Fig. 8.** The services for AMC as identified by SOAD [Zi04]

Having identified an initial set of web services, we stop at step 6 in the design method, given the limited information we have about the case. As in the [Zi04], the service model as such contains services only; it does not describe a functional system with control flow. Comparing our results with the results of SOAD (Fig. 8), we observe the following:

- The Inventory service (SOAD) corresponds to the Inventory Negotiation service.
- The Catalog service (SOAD) is an informational service that corresponds to the Garage Knowledge service.
- The Scheduling service (SOAD) schedules both bays and mechanics. It corresponds to the Work floor and Human Labor services. Whether it is preferable to combine these two, or keep them separated depends partly on whether they are always used in combination or not, and whether the responsibility for work floor and human resources is assigned to one or different managers, .
- The Work Order Service (SOAD) has no direct correspondence. Its function is to save the results of a negotiation (a work order) for the next phase of the car maintenance. In our model, the work order is provided by the Negotiation Service to the Car Maintenance Delivery Service of which it is a part.

- The Business Process (SOAD) corresponds to the Car Maintenance Negotiation Service (CMN). It is a composite service as it has to interact with other services on which it depends.
- The Customer service (SOAD) has not been identified in our model. Its function is to provide some information about the customer that the customer could provide himself as well (and would if he is visiting the garage for the first time). The Customer service may be viewed as an additional service of the garage to its regular customers – to maintain information about their cars. If so, it should be identified as an additional value transfer in the value model, and it could have given rise to an automated service as it is purely informational. Alternatively, the Customer service can be seen as another part of the delivery process, the Contacting that precedes the Negotiation proper and initiates an interaction between Garage and Customer. The results of this Contacting Service would be channeled to the Negotiation Service via the encompassing Car Maintenance Delivery (CMD) service. The case description is not detailed enough to decide.

From the comparison, we can conclude that our service design approach gives results that are compatible with the results of SOAD. It challenges the design choice made by SOAD to combine the reservation of mechanics and bays into one scheduling service and prompts for the need to be more specific about the Customer Service. In general, what can be seen as a positive feature of the method is that it is more specific about the type of service – in this case, that it explicates that the Inventory service and Scheduling service support negotiation processes (rather than information or evaluation processes). This may allow the designer to design these services as specializations of generic services types, for example for Negotiation, Information, Contacting, or Completion.

## 5. Conclusion

In this paper, we have developed a general (that is, not IT specific) conceptual service architecture. On the basis of this ontology, we have proposed a service design method that starts from a value model and identifies core and enhancing services and identifies possible web services. Whereas most SOA design methods consider service design as a software engineering problem, we consider it as both a business engineering and software engineering problem.

Service design such as outlined in this paper should be distinguished from service composition. The problem in the latter case is how to combine existing web services into a new composite service that meets specific requirements. Although there is some overlap (step 8), the objective of service design is to identify in a certain business context what are the economically viable services in the first place. On the other hand, service design should also be distinguished from exploring new value constellations, such as supported by e$^3$value: the former typically builds forth on the latter.

Starting off the service design with value models has the advantage that the analyst/designer can draw upon the available tools and methods in this domain, including recent extensions such as e$^3$services [KG08]. The latter paper focuses in particular on the customer needs and wants that motivate specific business services, which is particularly useful when innovative services are to be developed.

Topics for future research include validation by means of more cases, as well as the development or reuse of graphical notations supporting the service design.

## References

[An06] Andersson, B., et al Towards a Common Business Ontology. Proc. Interop-EMOI'06. Luxembourg (2006).

[Ba04] Z. Baida, J. Gordijn, B. Omelayenko and H. Akkermans. A Shared Service Terminology for Online Service Provisioning. In M. Janssen, H.G. Sol and R. W. Wagenaar editors, *Proc. of the 6th Int. Conf. on Electronic Commerce (ICEC04)*, ACM Press (2004).

[BP07] OASIS Web Services Business Process Execution Language Version 2.0 http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (2007).

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

[Fo97] Fowler M., *Analysis Patterns. Reusable Object Models.* Addison-Wesley, (1997).

[Ge99] Geerts, G., McCarthy, W. E. An Accounting Object Infrastructure For Knowledge-Based Enterprise Models. *IEEE Int. Systems & Their Applications*, pp. 89-94, (1999).

[Go00] Gordijn J., Akkermans J.M., van Vliet J.C.: Business modeling is not process modeling. In: *Conceptual modeling for e-business and the web*. LNCS, Vol. 1921. Springer-Verlag (2000).

[Go06] Goldkuhl, G. Action and media in interorganizational interaction. *Comm.. ACM* 49, 5 (May. 2006), pp.53-57.

[HJ07] Henkel, M., Johannesson, P., Perjons, E., and Zdravkovic, J.: Value and Goal Driven Design of E-Services. In *Proc. of the IEEE Int. Conference on E-Business Engineering (Icebe'07)*. IEEE Computer Society, Washington (2007)

[Hr06] Hruby, P. *Model-Driven Design of Software Applications with Business Patterns*. Springer Verlag ISBN: 3540301542. (2006).

[KG08] S. Kinderen, de and J. Gordijn. E3service: An ontological approach for deriving multi-supplier IT-service bundles from consumer needs. *Proc. HICSS*, IEEE (2008)

[Mc82] McCarthy W. E., The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *The Accounting Review* (1982).

[NL07] N Nayak, M Linehan et al. Core business architecture for a service-oriented enterprise. *IBM Systems Journal, 46*(4), pp.723-742 (2007)

[OA06] OASIS. Reference Model for Service Oriented Architecture 1.0. Available at http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf (2006)

[Os04] Osterwalder A. *The Business Model Ontology*, Ph.D. thesis (2004), HEC Lausanne. Available at http://www.hec.unil.ch/aosterwa/, last accessed 007-07-01.

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

[PH06] Papazoglou, M., & Heuvel, W.J.A.M. van den. Service-oriented design and development methodology. *Int. Journal of Web Engineering and Technology*, 2(4), 412-442. (2006)

[Pr04] C. Preist. A Conceptual Architecture for Semantic Web Services. In F. van Harmelen S.A. McIlraith, D. Plexousakis (eds.), *The Semantic Web ISWC 2004: Third International Semantic Web Conference*, Springer, LNCS 3298, (2004).

[UM03] UN/CEFACT Modelling Methodology (UMM) User Guide. Available at http://www.unece.org/cefact/umm/UMM_userguide_220606.pdf 2008-02-19. (2003)

[UN08] United Nations, Dept. of Economic and Social Affairs. Common DataBase (CDB) Data Dictionary. Available at http://unstats.un.org/unsd/cdbmeta/gesform.asp?getitem=398 2008-02-19.

[Us96] Uschold M., Gruninger M., Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 11(2), pp.93-155 (1996).

[Wi08] Wikipedia. Service (economics). (2008)

[WS04] W3C. *Web Services Architecture W3C Working Group*. http:www.w3.orgTRws-arch, (2004).

[WS05] Roman, D. et al. Web Service Modeling Ontology. *Applied Ontology,* Volume 1, Number 1 (2005).

[Zi04] Zimmerman, O., Krogdahl, P. & Gee, C., Elements of Service-Oriented Analysis and Design, www-128.ibm.com/developerworks/library/ws-soad1/ (2004)