Applied Ontology 0 (0) 1–0
IOS Press

e^3*service*: an ontology for needs-driven real-world service bundling in a multi-supplier setting

Sybren De Kinderen [a,*], Pieter De Leenheer [b,c], Jaap Gordijn [b], Hans Akkermans [b], Rose-Marie Dröes [d], and Franka Meiland [d]

[a] *University of Luxembourg, Luxembourg*
[b] *Department of Computer Science, VU University Amsterdam, The Netherlands*
[c] *Semantics Technology & Applications Research Lab, Vrije Universiteit Brussel, Belgium*
[d] *Department of Psychiatry, EMGO institute for health and care research, VU University medical center Amsterdam, The Netherlands*

Abstract.
Businesses increasingly offer their services electronically via the Web. Take for example an Internet Service Provider. An ISP offers a variety of services, including raw bandwidth, IP connectivity, and Domain Name resolution. Although in some cases a single service already satisfies a customer need, in many situations a customer need is so complex that a bundle of services is needed to satisfy the need, as with the ISP example. In principle, each service in a bundle can be provisioned by a different supplier. This paper proposes an ontology, *e*^{3}*service* , that can be used to formally capture customer needs, services, and multi-supplier service bundles of these. In addition, this paper contributes a process called PCM^2 to reason with the ontology. First, a customer need is identified for which desired consequences are elicited. Then, the desired set of consequences is matched with consequences associated with services. The matching process results in a service bundle, satisfying the customer need, containing services that each can be provided by different suppliers. PCM^2 is inspired by a family of formal reasoning methods called Propose-Critique-Modify (PCM). However, whereas PCM methods emphasize solution generation from a given set of requirements, our reasoning process treats the space of requirements as a first class citizen. Hence PCM^2: the requirements space and solution space are equally important. How the reasoning and matching process practically works, is illustrated by an industry strength case study in the healthcare domain.

Keywords: service value networks, ontology, problem solving method, customer need

1. Introduction

More than 80% of our global economy consists of services, and an increasing number of them is being traded electronically, i.e. online. Consider, for example, an Internet Service Provider (ISP), that may satisfy a customer's need to "communicate at a remote distance"' via a variety of alternative service offerings. These offerings may include basic services such as raw bandwidth, IP connectivity, and Domain Name resolution; but additional services, such as an email box, chat box, Voice over IP (VoIP), or Instant Messaging (IM) may be offered as well.

Although many customers may deal with just one ISP to have all their Internet-related needs satisfied, this is not always strictly necessary or possible. The above-mentioned basic ISP services may each be provided in their own right, and so by different suppliers. For example, the customer may obtain raw bandwidth from supplier S_1, IP connectivity and Domain Name resolution from supplier S_2, and an email box from yet another supplier. Partnering enables suppliers to cooperate and satisfy complex customer needs they never could have done on their own in a flexible way. This phenomenon is increasingly observed in e-commerce settings, especially when small enterprises come into play (see e.g., Tapscott et al. (2000)).

*Corresponding author: University of Luxembourg, 6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg-Kirchberg, email: sybren.dekinderen@uni.lu

Multi-supplier bundles often satisfy customer's need much better and cheaper, as opposed to a single supplier bundle. Moreover, fixed single supplier service offerings often contains unwanted services such as an ill-functioning spam filter of an email service with poor storage capabilities. Given this explosion of possible combinations, it is a technological challenge to provide automated support for composing the most optimal bundle in response to an individual's unique needs, and by doing so build long-term partnerships among the participating suppliers. The online nature of services requires *online ordering* by the customer, and online provisioning of those services. Therefore, ordering and provisioning processes should be facilitated by computational support insofar possible.

In this article, we introduce the *e*^{3}*service* ontology, that provides constructs to describe a 'customer need' and 'commercial service' in a machine-processable way. In so doing, we contribute an ontology, grounded in marketing literature, for capturing commercial services and customer needs. Additionally, the reasoning capability of the *e*^{3}*service* ontology allows to elicit a customer need, and to subsequently find a bundle of multi-supplier commercial services that satisfies the stated need. This reasoning process, called PCM^2, is actually a variation of the Problem Solving Method Propose-Critique-Modify(PCM). PCM refers to a family of formal reasoning methods that gradually arrive at a final solution by *iteratively* testing and adjusting candidate solutions Motta et al. (1996). As we explain in Sect. 6.2 'classic' PCM methods, such as discussed in Motta et al. (1996), hereby often treat the space of input requirements as a given. In contrast, however, our reasoning process treats the space of requirements as a first class citizen. Hence, this paper also contributes the reasoning method PCM^2, wherein the modification of the requirements space and solution space is equally important.

The work presented here is based on earlier published results (de Kinderen (2010); de Kinderen et al. (2009); de Kinderen et al. (2009); Kinderen and Gordijn (2008)).

It is important to understand that the *e*^{3}*service* ontology is different from ontologies in the field of software-based services, such as WSMO (see Fensel et al. (2006)) or WSDL (see Christensen et al. (2001)). The *e*^{3}*service* ontology focuses on the valuable aspects of *commercial* services, whereas software-based services focus on interface specifications to facilitate interoperability and orchestration between *software components*.

This article is structured as follows. In Sects. 2 and 3 we discuss the notion of "service" and "need-driven bundling", focusing on those aspects that are relevant for our work. Thereafter, in Sect. 4, we introduce our dementia-care case study. Subsequently the dementia case is used as a running case study in Sect. 5, where we introduce the concepts of the *e*^{3}*service* ontology, and in Sect. 6, where we present how to reason about service bundling with the *e*^{3}*service* ontology. Sect. 7 discusses software tool support for the *e*^{3}*service* ontology. Finally, Sect. 8 presents conclusions and suggestions for further research.

2. The Conception of Service in Service Science, Engineering, and Business Modelling

Maglio et al. Maglio et al. (2006) define Service Science as *"the study of the application of the resources controlled by one [service] system for the benefit of another [service] system in the context of an economic exchange"*. This study came along with a shift in marketing from goods-dominant to service-dominant logic Vargo and Lusch (2004) where *service* becomes this new unit of economic exchange. Hence, a *service* is conceived as a (value-providing or -integrating) action. This stands in strong contrast to goods-dominant logic in which a service is considered to be an object Ferrario and Guarino (2012). This paradigm shift was required to understand and develop new ways of value creation in networked enterprises Normann (1993).

In De Leenheer et al. (2013), we distinguish at least two areas of service study. They have been developing largely independently from each other, resulting in divergent service conceptions.

1. The *business perspective* aims to understand why enterprises should innovate and trade *real-world services* by considering value creation aspects. Yet most modelling approaches take an enterprise-centric perspective (e.g., McCarthy (1982); Osterwalder (2004)). As a consequence, they assume a

closed world[1] and their business service semantics, i.e., shared understanding of value aspects, remains *tacit* Polanyi (1966); hence unknown outside the organization. Also, these approaches classify a service as a *static* resource (i.e., endurant) rather than as an *occurrence* of actions (perdurant) in which resources are acted upon. Most *service network approaches* (see Razo-Zapata et al. (2011) for a survey) are process-based, hence focus on the problem of planning service execution. Not many network-centric approaches exist that are value-based and have automated support for representation and design of service bundles Razo-Zapata et al. (2012). Only recently SDL-compliant service meta-models have been proposed (e.g., Ferrario and Guarino (2012); Poels (2010)) that may lead to sound ontological foundations.

2. The *IT perspective* adopts service-oriented modelling as a paradigm for functional decomposition and engineering of distributed systems. Prominent service description meta-models (i.a., WSMO and WSDL) conceive a *software service* as a static function, and fail to convey any value creation aspect. Web service engineering aims at the interoperability of communication protocols (e.g., SOAP, REST) and data formats between heterogeneous "service parks" (see Erl (2004); and Sycara in Thomas and Fellmann (2007)). Process languages (BPMN, BPEL, etc.) are adopted for choreography, control flows, events, and temporal dependencies to define valid sequences of service invocations. It may be the case that some business decision logic is cryptically embedded within some complex control flow logic. However, this business decision logic is rarely made explicit. In the worst case, the business logic is largely hidden within expert components or deferred to some manual decision steps.

In this article, we take the first – i.e., the business – perspective on service conception. We focus on ontology-based methods for the automated componential design and representation of multi-party real-world service bundles in response to well-articulated needs. To mitigate the mismatch between desired and received bundle features, organizations increasingly disassemble their standardized offerings into their constituent parts, and then reassemble these into offerings that fit with specific customer needs (see Quinn and Paquette (1990); Normann (2000)). In the trading of commodity goods, a component-based approach is already well developed, as exemplified by Dell and Cisco. Yet in the service sector, a component-based approach for service bundling is still inadequate. We identify the following dimensions that likely could inhibit the description and design of real-world services in an automated way.

- *Variability* Real-world services have a high variability, in that they consist of many diverse elements. To illustrate, consider that a city transportation service can come in many varieties depending on the combination of physical resources that are available, or on what the service integrates or is willing to sacrifice. For example, a taxi service provides a highly comfortable but pricy transportation between two points in a city and merely requires a decent road infrastructure. A public transportation service, on the other hand, is less expensive but has only limited availability, hence may require additional sacrifice from the consumer to travel to and away from the transportation system and/or travel at a certain time.
- *Intangibility* Omitting the platform/infastructure and its physical properties that makes a service possible, the essence of service semantics boils down to the value being exchanged: i.e., value-in-use. As opposed to a good, that can be touched, weighted and transacted a service description is limited. Moreover, its value (such as comfort, sacrifice, price) can only be assessed during or after its consumption. This asks for a focus on the semantics of value as we will see next.
- *Co-production*. Many services are *collectively* produced by different peers. A commonly quoted aspect of services is that end-customers themselves often take part in their production Vargo and Lusch (2008); Karpen et al. (2012); Grönroos (2007).Zeithaml et al. (1985) refer to this as inseparability of production and consumption, i.e. "prosumption". As a result the true meaning of a service is manifested by the reciprocal relationships peers establish in order to coproduce that service. The networks that emerge from this are referred to as service value networks Razo-Zapata et al. (2011).

[1] A closed-world assumption is the logical presumption that what is not currently known to be true is false.

As a consequence, services are offered as a *bundle* consisting of *elementary* services. Service marketers define bundling as the practice of selling two or more products together in a single package (see Guiltinan (1987); Stremersch and Tellis (2002)). Examples include Internet (internet access, e-mail hosting, spam filters etcetera) and holiday (transportation, plus hotel, plus dinner-arrangements).

The design approach we aim for is in strong contrast with planning problems - typically solved with AI methods - such as the functional composition of control-flow elements and temporal dependencies to articulate the execution of (software / Web) services. These planning methods aim to answer *how* and *when* services exist; hence focus on the conception of events, states, and process. We are interested in *what* value is exchanged. Rather than describing the service itself, we focus on the conception of its value components in order to overcome the variability, intangibility, and collective nature of service descriptions.

3. Background for Needs-driven Service Bundling

Our approach, e^3*service* , uniquely exploits the tension field between problem structure, manifested by individual service needs; and the solution space, represented by the sum of all service descriptions. In the next section this will lead to the a dual perspective in the e^3*service* ontology of services. This strict separation between problem and solution is similarly found in the Requirements Engineering Field, as well as in marketing research as found in the commonly accepted Customer Buying Behavior (CBB) model Kotler (2000); Solomon (2003); Loudon and Della-Bitta (1993). As depicted in Fig. 1, the CBB model consists of five (not strictly consecutive) steps.

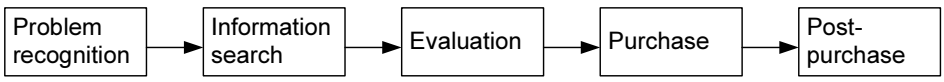

Fig. 1. The Customer Buying Behavior Model, cf. Kotler (2000)

The CBB model (Kotler, 2000, p. 177-178) provides a number of interesting mechanisms that helps us to understand better how service needs emerge; hence will give insights in how conceptual analysis has to be performed.

1. Steps (1) “Problem Recognition” and (2) “Information Search” emphasize the explicit separation between problem specification and problem solving. The recognition of a problem may be caused by a change in the customer’s current state (e.g. the desire for a new car because of a engine breakdown) or ideal state (e.g. by comparing her car to the new car of the neighbor). Once the problem is clearly identified, the customer searches for solutions satisfying his need through information search.
2. Step (2) “Information Search” suggests that customer needs analysis is treated as *problem specification*, whereby a high-level problem is articulated into a model specific enough for a solution to be found. *Means-end chaining* (see e.g. Gutman and Reynolds (1988, 1982); Gutman (1997)), which allows for such problem specification, links specific product attributes to high-level personal values via intermediary chains of *consequences*. The resulting *ladders* represent a conceptual map of how a customer group links specific product attributes to more static high-level values. For minty-breath, the ladder would for example be: “minty breath” (attribute) - “neat image” (consequence) - “increase social acceptance” (consequence) - “sense of belonging” (value), the latter subsuming “living a happy life”. Means-end chaining is particularly useful for discovering high-level motivations for a customer and is often used for positioning a product. Assume “living a healthy life” would be a more important personal value, then toothpaste would be positioned differently, with an emphasis on attributes such as “calcium” and consequences such as “strengthens teeth”. “Minty breath” would then not be so relevant. Means-end chaining has also been applied to various service domains including

internet Aschmoneit and Heitmann (2002), logistics Mentzer et al. (1997), railway Herrmann et al. (2000), enterprise architecture van der Raadt et al. (2008), and hotel accommodation Michael et al. (1999). A result of information search is usually a candidate set of service offerings satisfying the customer needs recognized in step (1).

3. Step (3) "Evaluation" reviews the candidate service offerings found in step (2). To this end, a service integrator seeks to find an optimal trade-off between positive consequences (added value) and negative consequences (sacrifice) of the service offered Gutman and Reynolds (1982). Solomon (2003) mentions different heuristics - commonsensical shortcuts - that customers tend to use to evaluate offerings. Interestingly, not all of these involve a comprehensive trade-off. Consider for example the lexicographical rule. With this rule, the customer first ranks product attributes according to their importance, after which any product that does not contain important attributes is not bought, irrespective of any other attributes this product might have. For example: Upon buying a computer, a customer that finds the feature 'Harddisk-size => 120GB' important, will not consider any computer that has a Harddisk-size $<$ 120 GB, irrespective of other features such as CPU-speed or the amount of internal memory.
4. Step (4) "Purchase" and (5) "Post-purchase" are not further considered as we focus on the design of services, not the value-in-use evaluation.

4. Case Study: Assisted Living Services for Dementia Patients

Study background. Since the privatization of the Dutch healthcare industry, informal carers of persons with dementia are granted a personal allowance by a general insurance fund to spend on healthcare services. The amount is determined by person-specific factors including age, income and marital status. The principle idea behind a personal allowance is that informal carers can select the healthcare providers that they are most comfortable with. These can be renowned physicians, or even friends or family members. Nevertheless, a personal allowance also introduces many challenges. First, many private and (semi-)public organizations offer variations of similar healthcare-services. As a result, healthcare practitioners cannot see the woods for the trees anymore when selecting services. Second, since informal carers generally lack expertise on healthcare services, they often have trouble expressing what they need in terms that are specific enough to find services for.

To deal with this abundant offering, Dröes et al. (2005) have developed a pilot Digital Interactive Social Chart for informal carers of persons with dementia, called DEM-DISC. The key idea behind this social chart is that informal carers can express their care needs by exploring an online question tree. Based on the tree traversal, the website responds with a list of appropriate service bundles. E.g., consider an informal carer who states that s/he is interested in social engagement for a person with dementia. Based on the postal code of that person, a "meeting group for people with dementia" service would be found that is in the vicinity of where the person with dementia lives. In this article, we apply $e^3service$ to improve the social chart, and report on feedback we got from healthcare practitioners. For the latter, this application is relevant because they are considering enhancing the reasoning capabilities of the current social chart. We explore alternative reasoning mechanisms, in particular by focusing on (1) customer-supplier interaction as an alternative to the currently used - projectionist - question specification tree. Here, "projectionist" refers to a one way specification from the customer's perspective on services to the supplier perspective; (2) gradual consideration of requirements; and (3) the application of both compensatory and non-compensatory decision models.

Study setup. The dementia-care case study consists of two phases. First, we instantiated the $e^3service$ ontologies for the healthcare domain and created a healthcare services catalogue. These instantiations were based on the existing social chart, supplemented with information gathered from interviews with domain experts (on, e.g., constraints such as pricing and waiting lists). Secondly, using the tool, we validated the usefulness of the $e^3service$ reasoning process with two 2+-hour tool demonstrations with the healthcare domain experts who were mainly responsible for creating the original social chart. Since the tool conforms

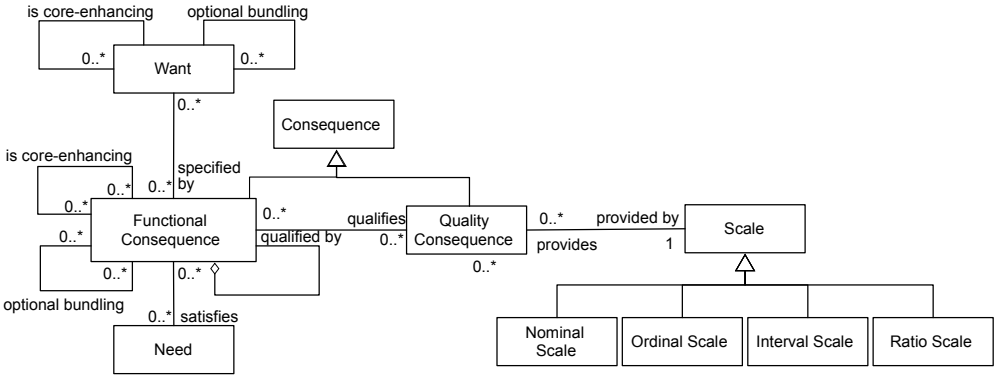

Fig. 2. The e^3*service* customer perspective ontology

exactly to the reasoning steps from e^3*service* , our assumption is that we can also validate the usefulness of e^3*service* through such a demonstration. The tool demo involved scenario walkthroughs, where realistic consumer needs - such as a customer that tries to find a meal-preparation service - were taken as starting points to show how the tool interacts with the informal carer. For each scenario walkthrough, the domain expert then commented to what extent the demonstrated principles could constitute a useful addition to the existing social chart.

5. The e^3*service* Ontology

The e^3*service* ontology takes two perspectives on service conception: the *customer* perspective (Sect. 5.1) and the *supplier* perspective (Sect. 5.2). Additionally, there is a *pricing* ontology (Sect. 5.3).

5.1. The customer perspective

The customer perspective ontology, depicted in Fig. 2, specifies the consumer's conception on the yielded value from consuming a service bundle. The ontology is based on concepts from *established* customer needs literature. Most notably, see Arndt (1978); Kotler (2000) for a discussion on needs wants and demands, Gutman and Reynolds (1982); Woodruff (1997); Gutman and Reynolds (1988); Gutman (1997) for a discussion on means-end chaining and Holbrook (1999) for a discussion on the different ways in which the value of a product/service can be represented. An instantiation of this ontology for our case study can be found in Fig. 3.

5.1.1. Concepts

Need A need is a basic human requirement that can be specified by means of one or more *consequences* Arndt (1978), Kotler (2000).

– *Example*: A commonly encountered problem stated by Dutch informal carers of persons with dementia is "I cannot cope anymore. What can help?".

Consequence A consequence is anything that results - directly or indirectly - from consuming a service Gutman and Reynolds (1982, 1988); Gutman (1997). It represents the effect of consuming a service Gutman (1997), and as such, expresses how service consumption aids (in-)directly in satisfying a customer need. Note that, being an effect of consuming a service, a consequence can be both an activity being brought about by service consumption (for example: the consequence 'Meal preparation') or a state (for example: the consequence 'social contacts dementia patient'). This is also in line with the definition and examples of a consequence proposed in Gutman and Reynolds (1982, 1988); Gutman (1997).

Functional Consequence. A functional consequence represents the functional goal that can be achieved through consumption of a service.

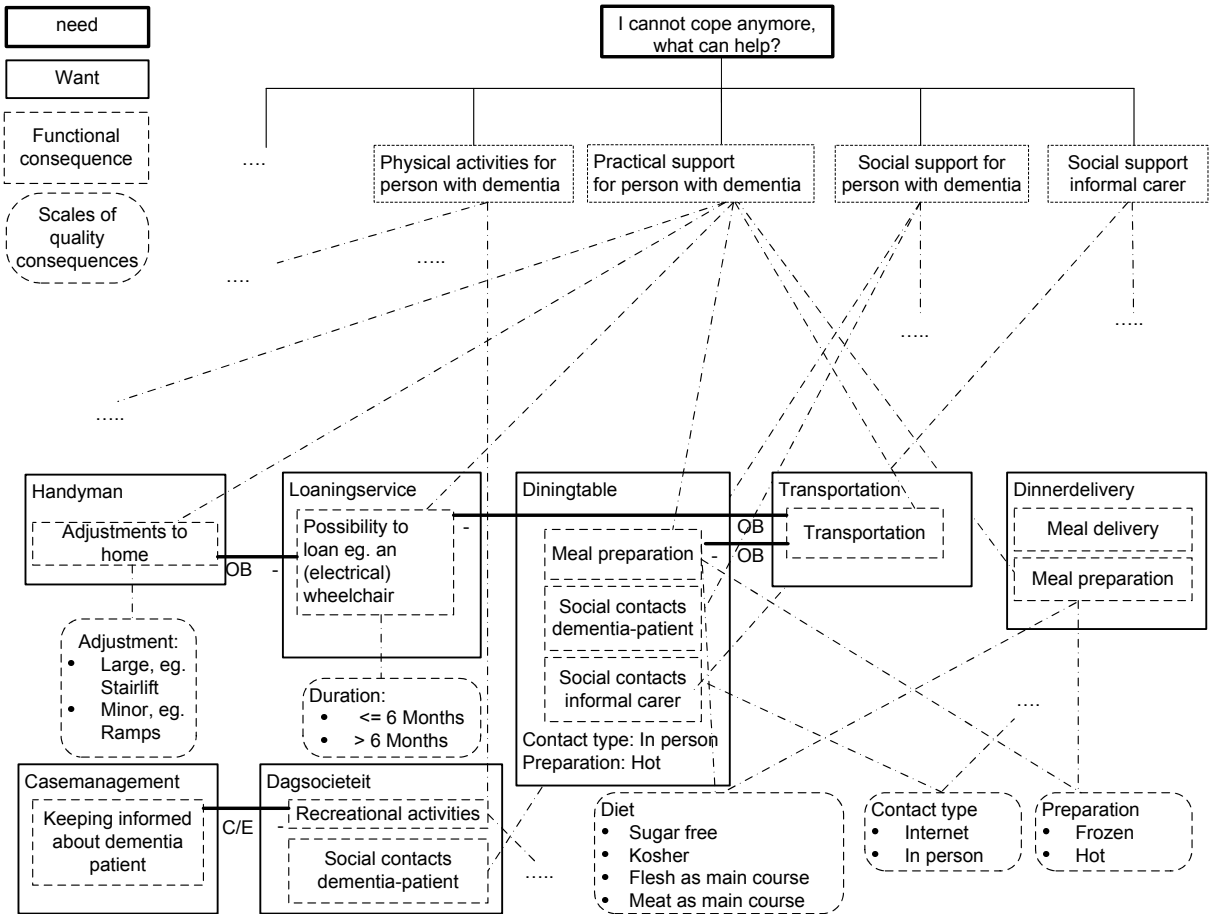

Fig. 3. Partial customer catalogue for the dementia case

– *Example:* The functional consequence 'Meal preparation'.

Quality Consequence. A quality consequence expresses qualitative properties of other, e.g. functional, consequences in customer terminology. Because it expresses the qualitative properties of a functional consequence, a quality consequence cannot be acquired separately. It is always *qualified by* (a relation between consequences explained below) a functional consequence.

– *Example:* The consequences 'Frozen' and 'Hot' are quality consequences for the functional consequence 'Meal preparation' (see the customer perspective catalog in Fig. 3). Note here that 'Meal preparation' is the primary function that interests the customer: 'Frozen' and 'Hot' express the qualitative properties that are only relevant *after* this primary function has been selected.

Want A want is a solution that at least one party is committed to provide Arndt (1978), Kotler (2000). The main difference between a need and a want: a need defines the essential "survival" set of consequences, while a want can contain non-essentials. Providers push them to the consumer. Wants, interpreted as supplier-independent solutions, can typically be found in (supplier independent)service taxonomies such as the NAPCS (see Ambler (1998)). In our e^3*service* ontology, wants act as a bootstrapping mechanism for consequences. This allows the customer to discover additional consequences that s/he may also be interested in.

– *Example:* A handyman, a Loaningservice.

Scale A scale (also used by software measurement, see e.g. García et al. (2006)) is used to group quality consequences of the same type.

We distinguish between four types of scales (adapted from García et al. (2006)), where each type adds expressivity to the previous: nominal, ordinal, interval, and ratio.
Nominal Scale A nominal scale indicates that a relationship exists between quality consequences, but introduces no ordering or ranking on these consequences.

– *Example:* a nominal scale "Diet" groups "Kosher" and "Glutenfree".

Ordinal An ordinal scale introduces an ordering on consequences such that it is possible to state that consequence X is better/greater than Y (but not *how much* it is better).

– *Example:* one may define a scale 'Meal size', with the quality consequences 'Small portion', 'Medium-sized portion', and 'Large portion'.

Interval An interval scale is an ordered scale where intervals between consequences at different points on the scale have the same meaning. This means that, as opposed to an ordinal scale, on a scale from 1 to 10, the difference between 2 and 3 means the same as the difference between 7 and 8.

– *Example:* The portions on the 'Meal size' scale may be measured in calories, which is relevant for the dietary needs of certain persons with dementia. We have an interval scale if the amount of calories in all subsequent portions is equidistant. For example, the portions may vary by 100 calories: a small portion may then be 300 calories, a medium-sized one 400 and a large portion 500 calories.

Ratio Additional to the properties of an interval scale, a ratio scale has an exact definition of a baseline measure.

– *Example:* Degrees Celsius is a unit measured on a ratio scale, because there exists an exact definition of the baseline (zero) measure, namely: When water transforms from a liquid to a solid form.

5.1.2. Relationships

The relations between Needs and Consequences:
satisfies Zero or more consequences satisfy a need.

– *Example:* Amongst others, the consequences 'Practical support for person with dementia', 'Social support for person with dementia' and 'Social support for informal carer' are ways to satisfy the need 'I cannot cope anymore. What can help?'.

is core-enhancing One or more consequences can be value-enhancing to a core consequence. As a prerequisite for acquiring the enhancing consequence, the core consequence must first be acquired.

– *Example:* The consequence 'Keeping informed about dementia patient' can add value to the consequence 'Recreational activities', provided that 'Recreational activities' is acquired first. This is because feedback for informal carers during case management is based upon observations made during the recreational activities of the person with dementia.

optional bundling An optional bundling relation between two consequences A and B indicates that consequence B can add value to consequence A, but that consequence B can also be acquired separately from A.

– *Example:* The consequence 'Adjustments to the home' can add value to the consequence 'Loaning of medical equipment'. To an informal carer, this combination of consequences makes sense because the loaning of medical equipment may imply impairment on the part of the person with dementia. As a result, adjustments to the home, such as the lowering of kitchen sinks or the installment of wall braces, may be a suitable consequence for the person with dementia.

qualifies A functional consequence qualifies zero or more quality consequences. This relation states that a quality consequence cannot be acquired without a functional consequence.

– *Example:* The quality consequences "Frozen" and "Hot" for the functional consequence "Meal preparation".

consists of A consequence may *consist of* one or more other consequences. Consequence *laddering* (see Gutman and Reynolds (1988)) can be used to specify abstract consequences into more concrete consequences until a sufficiently detailed consequence is found for which solutions can be offered.

– *Example:* The consequence "Practical support for person with dementia" consists of, amongst others, the consequences "Possibility to loan e.g. an electrical wheelchair", "Meal preparation", and "Transportation" (see the customer perspective service catalog in Fig. 3 for reference).

Relationships between Consequences and Wants:
specified by A want is *specified by* zero or more consequences. These consequences are used to state *how* the want, being something that can be provisioned commercially by a single supplier, satisfies a need. Note here that the *specified by* relation enables the bootstrapping of consequences, as shown in the example below.

– *Example:* Consider the want 'Diningtable', whereby the idea is to provide both a social and practical function by letting multiple persons with dementia and their informal carers dine together at a care home. The relation 'specified by' is used to demonstrate that a Diningtable has multiple valuable outcomes: it is *specified by* the consequence 'Meal preparation', as well as the consequences 'Social support for person with dementia', and 'Social support for informal carer' (for reference, see the customer perspective catalog in Fig. 3).

So, with the example of the dining table, we see that a want can bootstrap consequences that are perhaps also of interest to the customer. For example: an informal carer that is interested in 'Meal preparation', may find out that through selecting a Diningtable s/he receives social support as well. This illustrates how the range of consequences initially specified by a need can be extended when being exposed to wants.
is core-enhancing A want is core-enhancing for zero or more other wants. The Core-Enhancing relationship is used to state that for a certain want A, *provided that want A is acquired*, there exist zero or more wants B that could add value to A. The Core Enhancing relationship also exists on the level of consequences. However, this relationship also exists on the level of wants as wants actually package sets of consequences available on the market.

– *Example:* The value-enhancing want 'Casemanagement' is in a Core-Enhancing relationship with the basic want 'Dagsocieteit' (note: the idea behind a dagsocieteit is that it provides persons with dementia with daycare in informal surroundings. It includes amenities such as recreational activities, hot meals, and more). This is because there is no business logic behind having 'Casemanagement' for the informal carer without feedback from the participation of the person with dementia in the 'Dagsocieteit'. Note that the Core Enhancing relationship between consequences (discussed previously) is used to indicate *why* the want 'Casemanagement' adds value to the want 'Dagsocieteit': the person with dementia has to participate in 'Recreational activities' from 'Dagsocieteit', as a prerequisite for 'Keeping informed about dementia patient' during 'Casemanagement'.

optional bundling A want can be optionally bundled with zero or more other wants. It is used to state that for a certain want A, there exist zero or more value-enhancing wants B. However, as opposed to the core-enhancing relationship, want B can also be acquired separately from want A.

– *Example:* the want 'Handyman' can add value to the want 'Loaningservice'. Similar to the Core Enhancing relationship, the Optionally bundled relationship between two consequences can be used to show *why* the want 'Handyman' adds value to the want 'Loaningservice' (i.e. through the value-enhancing consequence 'Adjustments to the home').

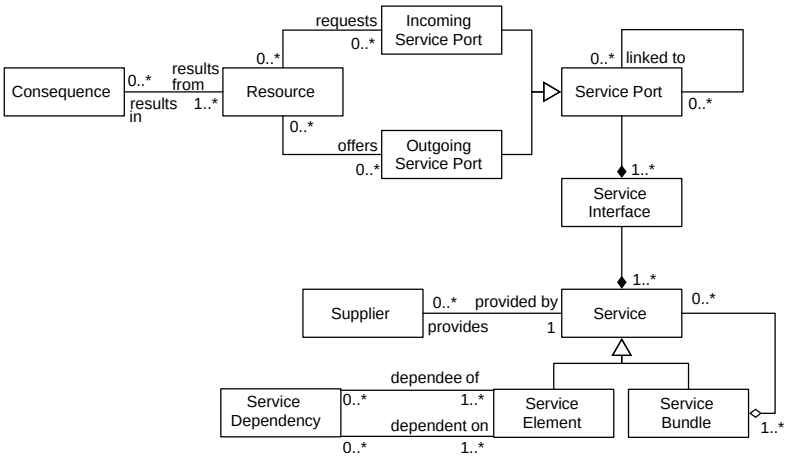

Fig. 4. The $e^3service$ supplier perspective ontology

5.2. The supplier perspective

The supplier perspective ontology, depicted in Fig. 4, specifies the supplier's conception on the anticipated value for a service bundle. It is derived from Akkermans et al. (2004) with the addition of one key concept from the consumer perspective: a consequence. The consequence forms the bridge between customer and supplier for the reasoning process.

5.2.1. Concepts

Service. A service is a (composed) activity, of particular importance to us being the created value for its environment.

Service Element. A service element is an activity that creates the service. Service elements are *elementary*, meaning that they cannot be decomposed into smaller elements that can be provided commercially.

– *Example:* The service element Diningtable (see Fig. 5).

Service Bundle A service bundle groups (multiple) service elements.

– *Example:* {Diningtable,Transportation}. Here, the customer requires the transportation service to get to the location of the dining table service.

Resource A resource is a unit of delivery that can be provisioned in its own right and in a commercially feasible way. In our ontology, a resource is characterized as a *static* in- or output. This is opposed to our conception of a service, which is characterized by being an activity.

– *Example:* Consider the example Diningtable service from Fig. 5. Here, the service element 'Diningtable' produces the resource 'Meal', and requires as input the resources 'Fee' and 'Transport'.

Service Port. Resources are provided or consumed by service elements via *service ports*, represented as arrowheads in Fig. 5. Service ports are a means of abstracting away from the inner workings of a service element so that one can focus on modelling the (valuable) inputs and outputs of a service element. We distinguish between *incoming* and *outgoing* service ports: Resources that are required as input for a service element are attached to an incoming service port of that element, whereas resources that are produced as output of a service element are attached to an outgoing port of that element.

– *Example:* The resource 'Fee' is required input for the service element 'Diningtable', thus attached to an incoming service port. 'Diningtable' in turn, provides the resource 'Meal' to the customer, thus 'Meal' is attached to an outgoing service port (where consequences such as 'Meal preparation' and 'Social contacts informal carer' show *why* this resource, and the properties it contains, is valuable to the customer).

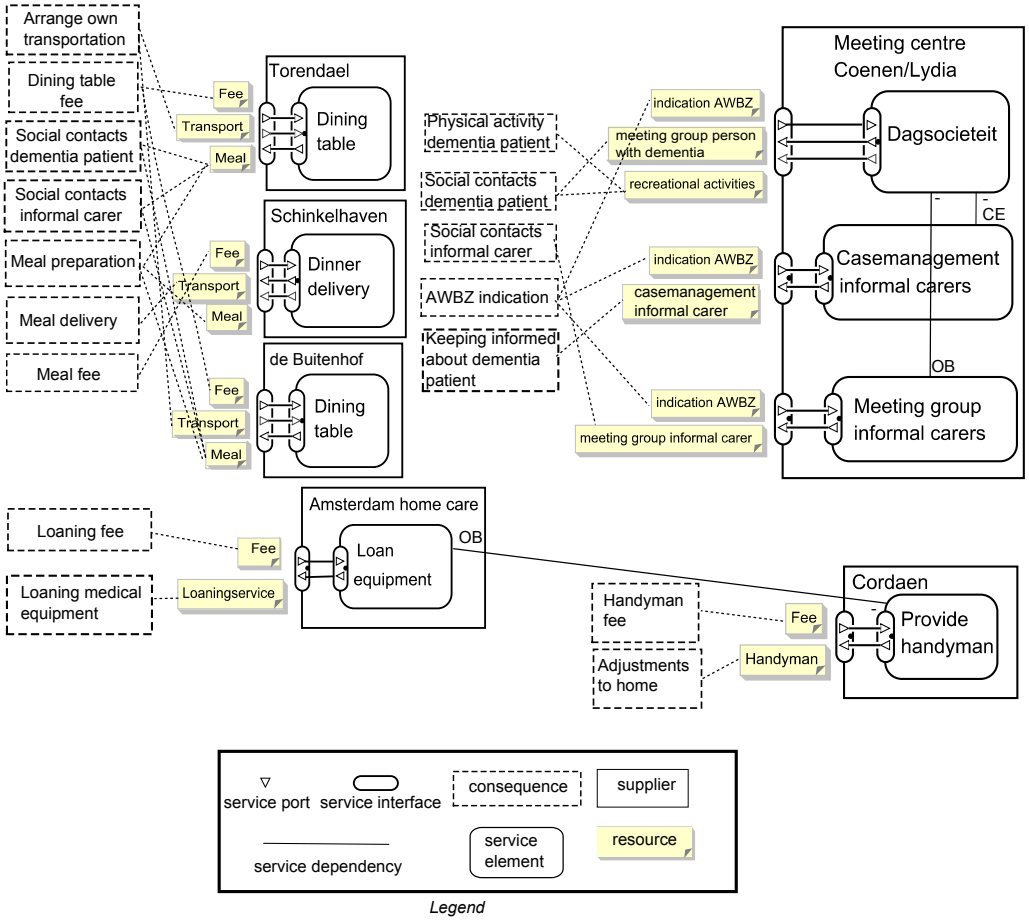

Fig. 5. Individual supply-side dementia care services

Service Dependency. Service dependencies are used to model dependencies between service elements (Akkermans et al. (2004)). These dependencies are unidirectional, meaning that the dependency between services A and B exists *independently* of that between B and A. In the supply-side catalog we find two dependencies (Fig. 5):

- S_1 is in a *Core/Enhancing* relationship with S_2 if (1) S_2 possibly adds value to S_1, (2) acquisition of S_1 is obligatory for acquisition of S_2 and (3) S_1 can be acquired separately from S_2. Notation: C/E.
  EXAMPLE: There exists a C/E dependency between 'Dagsocieteit' and 'Casemanagement'. As explained in the customer perspective ontology, case management is only relevant if the informal carer receives feedback from participation of the person with dementia in the recreational activities of the 'Dagsocieteit'.
- S_1 is in a *Optionally Bundled* relationship with S_2 when (1) S_2 possibly adds value to S_1 and (2) S_1 and S_2 can be acquired separately. Note here that, as opposed to the C/E relationship, S_1 does not have to be acquired before S_2 can be acquired. Notation: OB.
  EXAMPLE: There exists an OB dependency between 'Loan equipment' (Amsterdam home care) and 'Provide handyman' (Cordaen). As explained in the customer perspective ontology, a handyman may be useful to make adjustments to the home to go along with the loaning of medical equipment.

Service interface. A service interface groups one or more service ports. It enforces that *all* resources attached to the service ports in this interface must be exchanged, or *none at all*. Hence the composition relation between service interface and service ports in the supplier ontology (Fig. 4): if a service interface ceases to exist, the service ports that it groups cease to exist as well. The interface models economic

reciprocity or in other words: one good turn deserves another. Importantly, the grouping of resources attached to the service ports in the service interface shows bundling, in that all of the resources attached to the ports must be exchanged.
Supplier. A supplier is an actor, often a legal and profit and loss responsible entity.

5.2.2. Relations

results from A consequence *results from* one or more resources.

- *Example:* The consequence 'Arrange own transportation' results from the resource 'Transport'. This resource has to be sacrificed by the informal carer to acquire the 'Diningtable'.

offers,requests An outgoing service port offers zero or more resources, while an incoming service port requests zero or more resources.

- *Example:* The resource 'Meal' is offered by the outgoing port of 'Diningtable' (de Buitenhof). Note here that "de Buitenhof" is the name of the provider of the Diningtable.

part of A service interface is part of one or more services.
part of one or more value ports are part of a service interface.

- *Example:* For the service interface of 'Diningtable', 'Meal' is offered via an outgoing port, and a 'Fee' and 'Transport' are requested via an incoming port.

provides A supplier provides an elementary service.

- *Example:* The care home 'Torendael' provides the service element 'Diningtable'.

5.3. The pricing model ontology

According to Guiltinan (1987), a service bundle is a set of services that are sold together at a single price. E.g. for a fixed monthly fee, a mobile phone subscription allows a customer to call 200 minutes for free and send 100 short messages each month. For each extra message sent or minute called, s/he pays an additional usage-based fee. The *single* end-price is calculated, partly based on a fixed fee, partly on a usage-based fee.

In this pricing scheme example, calculating an end price is straightforward. However, pricing schemes can become complicated easily by introducing conditions on the prices set. E.g.,, the flat fee for 200 free calling minutes may only hold for the first three months. Thereafter, it might be reduced to 100 minutes per month. It could also be that the free calling minutes are only valid at off-peak times.

The service pricing literature (see Monroe (1990); Guiltinan (1987); Tung et al. (1997)) unilaterally states: no matter how complicated the setting of a bundle-price is, we can assume that the *resulting* set price is eventually based on a few basic concepts:

1. the price of an individual service can be fixed, usage-based or a combination of both;
2. the price of a service bundle always consists of the prices of the individual services contained in that bundle;
3. a discount might apply to the prices of the individual services when bundled;
4. a certain condition might hold on when to apply a certain pricing scheme, e.g. a certain discount.

Conceptualizing pricing models allows for a *uniform* way of representing the price setting of a bundle. This enhances human understanding, but more importantly these conceptualisations are formatted in computer-processable way. Hence, bundles can be compared at a large scale simply by letting the customer fill in some variables, such as the amount of minutes called to mobile lines each month.

Based on the four concepts above and a literature survey on pricing models Miranda et al. (2006), we conceptualised two pricing models categories (1) pricing models of elementary services, and (2) pricing models of bundles of elementary services. The resulting ontology is depicted in Fig. 6.

5.3.1. Concepts

Elementary pricing model.

We distinguish four types of pricing models for elementary services: Flat fee, Usage based, Two-part tariff, N-block tariff.

Flat Fee The customer pays a fixed fee for a service, *independently* of usage.

– *Example:* Monthly health insurance fee, which has to be paid independently of whether healthcare services are used.

Usage based. The customer pays a fee only for using (part of) a service. Thus, the fee paid by the customer is *fully dependent* on usage of a service.

– *Example:* A fee of €7 for using a dining table service.

Two-part tariff. The customer pays a fixed fee plus a usage based fee for a service.

– *Example:* Specialized transportation. Usually, the pricing of specialized transportation works just as a taxi: the total fare is a result of a base fee, plus a usage-based fee that is calculated from the driving distance.

N-block tariff. This pricing model consists of N usage-based pricing models. For each block the usage entity is the same, but the usage amount paid per entity differs. For example in case n=2, that is, 2 usage-based blocks, as long as a certain condition holds, say c1, the user of a service pays price P1 per usage entity X for block 1. Then, for a second condition defined on block 2, the user pays price P2 per usage entity X.

– *Example:* On the electricity market, you pay price P1 per kWh for hour 1, price P2 for hour 2 and so on. So, if you want to calculate the price of electricity for 3 hours, and the prices per kWh are €0.034, €0.038, €0.032 for hours 1,2,3 respectively, then the calculation would be: $X_1$0.034 + $X_2$0.038 + $X_3$0.032 (X_n being the kWh of electricity used in a certain hour).

Discount. A discount applied to a pricing model, usually in terms of a percentage of the final price.

Bundle pricing model.

The pricing model of a service bundle consists of the pricing models of the elementary services in this bundle, plus one or more discounts on the outcome of these pricing models. In accordance with Monroe (1990), we apply discounts to bundles in two ways:

(1) A *single* discount is applied to the outcome of the pricing models of all elementary services in a bundle.

– *Example:* For the sake of example, consider the bundle {Diningtable (Torendael), Transport (Stadsmobiel BV)} with a single discount of 10%. For this bundle, the calculation is 90 %($PM_{Diningtable}$ + $PM_{Transport}$). For one return trip to the dining table of 12 kilometers, the calculation would then be 90%(€7 + (2.50 + €0.25 * 12)) = €11.25.

(2) The pricing model of a bundle is an aggregate of the discounts applied to the pricing models of *elementary* services. We depict this in our ontology by aggregating one or more 'Pricing Model' in a 'Bundle Pricing Model'.

– *Example:* We again define the bundle {Diningtable (Torendael), Transport (Stadsmobiel BV)}, but now with a 15 % discount only on the pricing model of the elementary service 'Diningtable' (Torendael). The calculation would then be: 85% ($PM_{Diningtable}$) + PM$_{Transport}$. So in case our customer makes a return trip to the dining table of in total 12 kilometers, the calculation would be 85%(€7)+ €0.25 * 12 = €8.95 .

Formula. A formula expresses a pricing model mathematically.

– *Example:* For the n-block pricing model example, the formula is $X_1$0.034 + $X_2$0.038 + $X_3$0.032.

5.3.2. Relations

belongs to As a service interface represents bundling, the 'Bundle Pricing Model' belongs to one 'Service Interface'. An 'Elementary Pricing Model', on the other hand, is attached to one 'Service Port' of its service element.

defines A pricing model defines its corresponding formula for price calculation.

applies to A discount applies to zero or more pricing models.

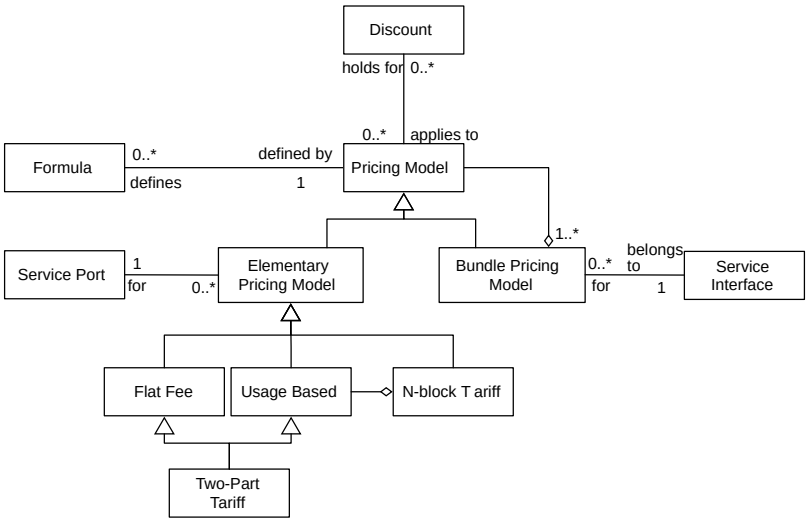

Fig. 6. The pricing model ontology

6. Reasoning with the *e*^{3}*service* Ontology

In this section, we apply the *e*^{3}*service* ontology to (semi-)automatically reason about needs-driven service bundling. First, in Sect. 6.1, we populate a service candidate space by generating all possible service bundles including bundling dependencies introduced earlier in Akkermans et al. (2004). We do this for two scenarios. Thereafter, we provide a high-level overview of the *e*^{3}*service* reasoning process and characterize it as a Propose-Critique-Modify (PCM) problem solving method (in Sect. 6.2). Finally, we discuss in detail the reasoning process that matches needs with the candidate space (in Sect. 6.3).

6.1. Generating the Service Candidate Space

We demonstrate the *e*^{3}*service* reasoning process by means of a scenario-walkthrough for two typical needs of informal carers of persons with dementia. The first scenario concerns finding a meal preparation service for a person suffering dementia, in order to demonstrate our customer-supplier interaction method. Our second scenario concerns the need for a loan for medical equipment, such as an electric wheelchair. With this scenario, we show how we create interesting combinations of services through demand interdependency. For reference, the case-specific knowledge can be found in the customer catalogue in Fig. 3 and the candidate space in Fig. 7.

To generate the service bundles, we rely on the supplier-perspective service dependencies discussed in Sect. 5.2. For example, by applying the supply side service dependency O/B (Dagsocieteit (Coenen/Lydia), Casemanagement informal carer(Coenen/Lydia)) modeled in the supply side catalog of dementia services (Fig. 5), we pregenerate two possible service bundles: {'Dagsocieteit' (Coenen/Lydia)} and {'Dagsocieteit' (Coenen/Lydia), 'Casemanagement informal carer' (Coenen/Lydia)}. The latter bundle is also present in our supply-side catalog of pregenerated dementia care service bundles (see Fig. 7).

Obviously, exhaustively generating all possible service bundles is not the best computational choice. Ideally, service bundles should be generated *dynamically*, such that only those service bundles that satisfy a specific customer need – and are actually offered – are in the candidate space. However, as the focus of this article is the matching of articulated needs with existing services, we omit this problem from this paper.

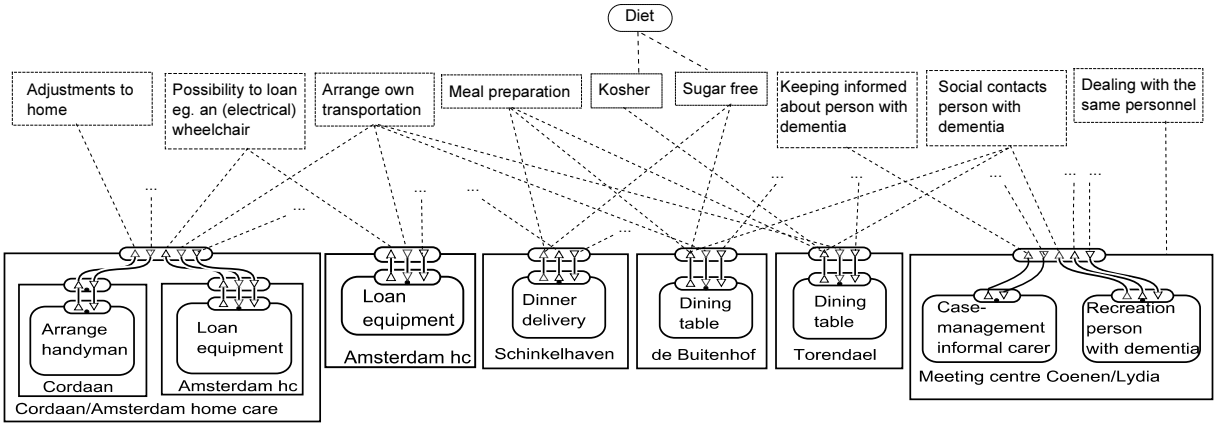

Fig. 7. Sample of the generated service bundles that constitute the candidate space (omitting the dependencies).

6.2. PCM^2: *The* e^{3}service *reasoning steps*

Fig. 8 presents the *e*^{3}*service* high-level reasoning process.
This reasoning process is a variation of the problem solving method Propose-Critique-Modify (PCM). PCM is defined by a commonly occurring combination of individual reasoning steps (referred to by Schreiber et al. (2000) as inference tasks): Propose, Verify, Critique and Modify. Although the exact order of these reasoning steps may vary, all PCM-based methods share one central premise Motta (1999),Chandrasekaran (1990): they *gradually* arrive at the final artifact by testing proposed solutions (usually against a fixed set of constraints), and revising these solutions if they are unsatisfactory (usually, solutions are unsatisfactory if they violate a set of constraints provided upfront).

This section discusses the *e*^{3}*service* reasoning process according to the aforementioned PCM inferencing tasks (in Sect. 6.2.1), and subsequently compares this reasoning process to other PCM methods (in Sect. 6.2.2).

6.2.1. *The* e^{3}service *reasoning steps as PCM inferencing tasks*

Propose

Task Description: Generate an initial partial or complete solution. Exit in case of failure to find a solution.
Domain specific implementation: Specify customer consequences, find and rank service bundles, present ranked bundles to customer.

The customer starts the reasoning process by selecting a need. In our running example, a need may be 'I cannot cope anymore. What can help?' (for reference, see the customer perspective catalog of dementia-care services in Fig. 3).

Subsequently, through a smart dialog with the customer, 'choose consequences'(see Fig. 8) specifies the need into consequences that are specific enough to find services for. Next, these desired consequences form input for the step 'Find service bundles'. For our running example we assume that, via our dialog, the customer selects the detailed set of consequences {'Meal preparation', 'Social contacts person with dementia', 'Social contacts informal carer'} which our reasoning process matches to the bundles {Diningtable (de Buitenhof)} and {Diningtable(Torendael)}. Finally, these bundles are proposed to the cus-

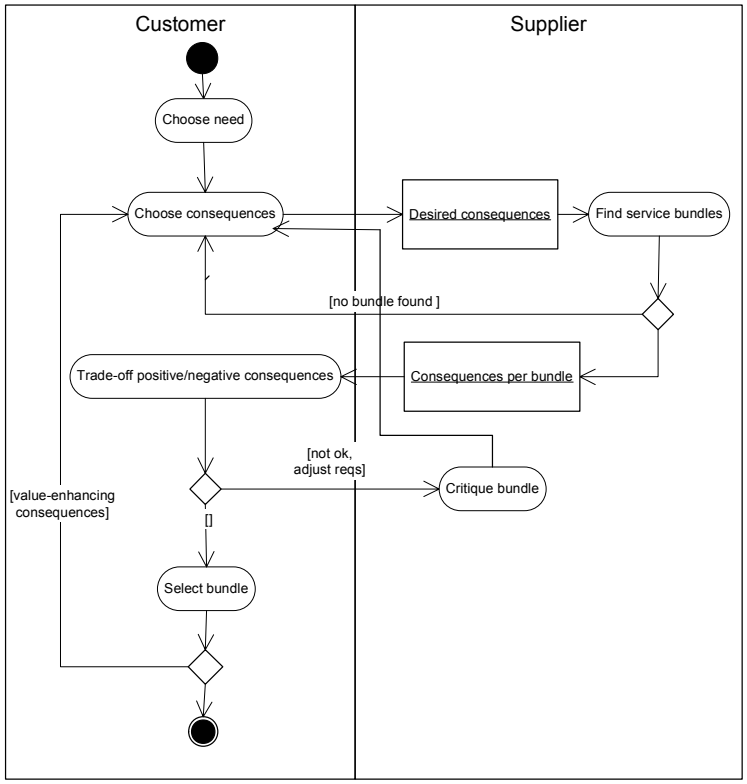

Fig. 8. The generic reasoning structure of e^3*service*

Positive consequences	Priority	Negative consequences	Priority
Meal preparation	Must have	Dining table fee	Reasonable
Social contacts informal carer	Must have	Arrange own transportation	Won't have
Social contacts person with dementia	Must have	-	-

Table 1

Trading off positive and negative consequences for the bundles {Diningtable (Torendael)}, {Diningtable (de Buitenhof)}

tomer. Note that the smart dialog and subsequent matching are described in further detail in Sects. 6.3.1 - 6.3.4.

Verify

Task Description: Checks if the proposed solution satisfies the required properties.
Domain specific implementation: The customer evaluates service bundles by trading off positive and negative consequences. Bundles initially found can be rejected due to the costs incurred.

The customer is presented with the found bundles and their respective consequences. Per bundle, the customer weights benefits and sacrifices (indicated by 'trade-off positive/negative consequences' in Fig. 8).

For the running example, a trade-off for the proposed bundle can be found in table 1. Here we see that the customer is not satisfied with the transportation that has to be arranged for the bundle (as is indicated by the qualitative label 'won't have' in table 1[2]). As a result the bundles {Diningtable (Torendael)} and {Diningtable (de Buitenhof)} will be rejected because they contain a negative consequence that the customer definitely wants to avoid: 'Arrange own transportation' (see Sect. 6.3.5 for further details).

[2] 'won't have' is a simplified label the sake of example: as we will see later on, we use quantitative importance scores from 1 to 10 to indicate importance of a consequence.

Critique

Task description: When a solution fails to meet the required properties, the critique task identifies the sources of failure.

Domain specific implementation: Our reasoning process identifies exactly *what* consequence (or set thereof) is responsible for the undesired sacrifices.

For our running example, the reasoning process finds the culprit by first tracing this consequence to the two 'Transport' resources. Subsequently, it finds the service ports to which these resources are attached (the black arrowheads in Fig. 9). In turn, these ports are traced to the service port of the *individual* service elements that require 'Transport' as input: 'Diningtable' from the supplier 'Torendael', and 'Diningtable' from the supplier 'de Buitenhof'. Finally, for each service element, the full set of consequences can be found through its service interface (the ovals encompassing the service ports in Fig. 9). So: from the critique step we learn that to acquire a diningtable service element with the consequences {'Meal preparation', 'Social contacts person with dementia', 'Social contacts informal carer'} the informal carer has to sacrifice the consequence 'Arrange own transportation'.

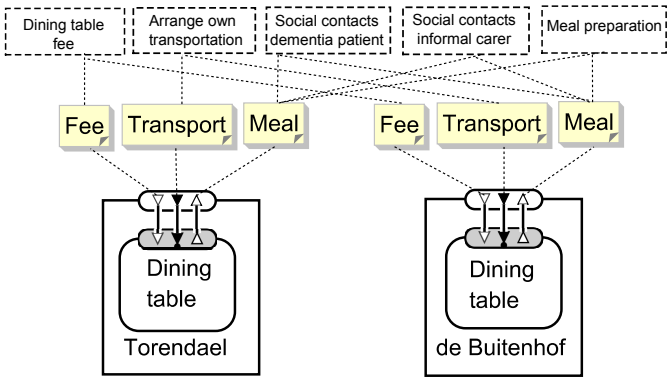

Fig. 9. Dementia care bundles. The critique part, where costs sources are identified, is highlighted in grey (for the service interface) and black (for the service port).

Modify

Task Description: Alternative values are filled in for the requirements that the critique-step identified as sources of failure. Adjusted solutions are found, which are again verified and if needed, criticized and modified until an acceptable solution is found.

Domain specific implementation: The customer modifies the initial consequences, taking into account the feedback from the critique-step. For the modified consequences, service bundles are again found and verified by the customer. If needed, the found bundles are criticized and modified again until a service bundle has been found that satisfies the desired consequences, against acceptable costs.

In our scenario, the customer returns to the step 'choose consequences' to modify the initially desired consequences. We assume that the customer now chooses the consequences from the want Dinnerdelivery. As such, s/he keeps the consequence 'Meal preparation', but leaves out the consequences 'Social contacts person with dementia' and 'Social contacts informal carer' implied by the want Diningtable. Taking the modified set of consequences as input, our reasoning process again generates service bundles. A possible outcome is the bundle presented in Fig. 10: Dinnerdelivery (Schinkelhaven). This bundle provides the consequence 'Meal preparation', but without having the negatively valued consequence 'Arrange own transportation'.

The customer may now decide to acquire a bundle and terminate the reasoning process. Alternatively however, s/he may decide to proceed by scoring value-enhancing consequences.

PCM for value-enhancing consequences When the customer chooses to expand the service offering

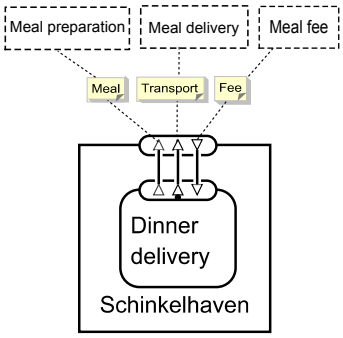

Fig. 10. Offered service bundle, based upon revised consequences

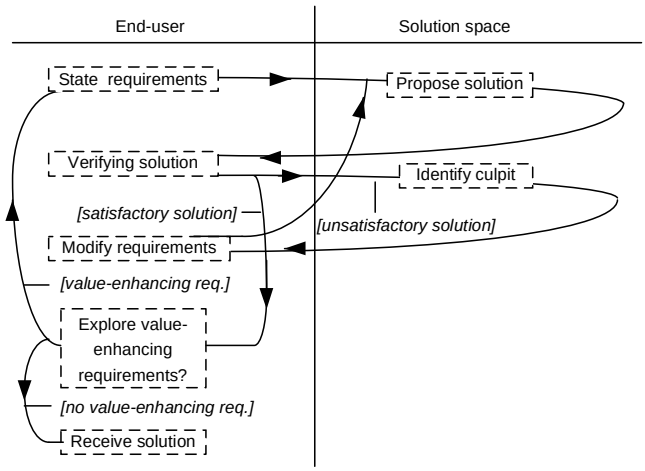

Fig. 11. The control structure of PCM^2

with value-enhancing services, *e*^{3}*service* relies on the PCM steps presented thusfar. However, it now uses value-enhancing consequences as input (indicated by the guard [value-enhancing consequences] in Fig. 8). For example: for the consequence 'meal preparation' from the selected dining table, a value-enhancing consequence is 'transportation'. Subsequently, through our PCM reasoning process, a fitting transportation service is found to supplement the dining table.

6.2.2. PCM^2 compared to PCM methods

It is important to point out that the *e*^{3}*service* reasoning process is inspired by PCM, but that it cannot be considered as a 'classic' PCM-based method. In classic PCM-based methods, such as Complete-Model-Then-Revise Motta (1999), the emphasis lies on finding solutions, while constraints are defined upfront and remain unchanged during the reasoning process. In contrast the *e*^{3}*service* reasoning process treats the space of input requirements as a first-class citizen, wherein constraints are also subject to change. For this reason, we refer to the *e*^{3}*service* reasoning process as a PCM^2 problem solving method: the requirements and solution space are equally important.

In Fig. 11, we present the control structure of PCM^2. Here, we see that both requirements and solutions are subject to change in the *e*^{3}*service* reasoning process. This is largely due to:

- The trade-off that a customer has to make between the positive and negative consequences of a bundle. We have seen that negative consequences, such as a price to be paid or customer lock-in, can modify the consequences initially desired;
- The incremental creation of a service bundle. We have seen that the customer first agrees on a bundle that provides the basic consequences, and only thereafter considers the value-enhancing consequences.

Thus, we deal with requirements *dynamically*: As the reasoning progresses, new requirements are considered, requirements are prioritized differently and requirements may be removed altogether. We call this approach PCM^2, to reflect that not only solutions can be subject to modification, but requirements also.

If we compare PCM^2 to other PCM-based methods in more detail, we find two major differences: in the definition of constraints, and in the emphasis on solutions.

- *PCM-based methods assume constraints to be defined upfront, and to remain unchanged over the course of the reasoning process* To the best of our knowledge, PCM-based methods (such as Poeck et al. (1996),Grimnes and Aamodt (1996),ten Teije and van Harmelen (2004),Schreiber and Terpstra (1996)) use a set of unchanging constraints to trigger modifications. These constraints are assumed to be a static given at the start of the reasoning process.
  In contrast, PCM^2 does not consider constraints to be a static given, because the constraints for end-customers are *relative*. As the reasoning progresses, new constraints may be discovered, and existing ones may be reprioritized or even removed altogether.
- *PCM-based methods modify requirements, but do so mostly in the light of solutions* We have found that PCM-based methods may also modify requirements, but that the emphasis in doing so always lies in a computational argument for finding solutions. For example, consider the comparison of the control structures extend-model-then-revise and complete-model-then-revise in Motta et al. (1996). Here, the arguments of choosing one control structure over another focus mostly on solutions, such as complete-model-then-revise being able to minimize the amount of constraint violations. However they ignore that requirements may play an important part in PCM-based methods as well. In PCM^2 for example, an important argument for using extend-model-then-revise is that it provides the user with a 'smart' way of traversing the space of requirements.

Finally, please note that Chandrasekaran (1990) and ten Teije et al. (1998) already point out that both requirements and constraints may be subject to modification also. Yet, even the above references do not capitalize on this notion beyond an observation. Therefore, to the best of our knowledge, PCM^2 is the first method that really emphasizes that requirements and constraints are just as much subject to change as solutions are.

6.3. Detailed reasoning steps

We now present the *e*^{3}*service* reasoning process in more detail. For illustration purposes, the reasoning process is accompanied by tool output for the dementia-care case.

6.3.1. Choose need and choose consequences

Reasoning steps. Starting from a customer need, we derive an initial set of consequences satisfying this need. In the customer perspective ontology (Fig. 2), we do this by expanding the relation 'satisfies' from a single need to one or more consequences specifying this need. First we focus on functional consequences only, since a functional consequence shows *how* a service can satisfy a need. The reasoning process first asks the customer to choose a particular consequence and then checks whether the selected consequence 'consists of' other, more detailed, consequences. If so, the customer is again asked to make a choice after which, for all chosen and implied consequences, the reasoning process again reviews whether considered consequences 'consist of' other consequences. This continues until no more 'consist of' relationships are found.

Next, the *e*^{3}*service* reasoning process derives one or more wants that contain this consequence. Here, the assumption is that the experts that created the service catalog as used for the reasoning process have defined solutions upfront for detailed (i.e. leaf) consequences. Subsequently, the reasoning mechanism derives additional consequences that are also part of the want. In other words: the want acts here as a bootstrapping mechanism for consequences (see case example).

You will be shown the available list of needs. Please choose one:
1: I cannot cope anymore, what can help? 1
We have the following consequences:
1: Physical reactivation
2: Social support informal carer
3: Social support person with dementia
4: Practical support person with dementia
Please choose a (group of) consequences: 4
1. The want Handyman satisfies the consequence Practical support for dementia patient through the consequence 'Adjustments to home'
2. The want Diningtable satisfies the consequence Practical support for dementia patient through the consequence 'Meal preparation'
Diningtable also has the following consequences:
*Social contacts for person with dementia, Social contacts for informal carer
Finally: in choosing this group of consequences, you will automatically be assigned the following quality consequence:
* Presentation: Warm for the functional consequence Meal preparation
* Contact medium: in person for the functional consequence social contacts informal carer
3. The want Dinnerdelivery satisfies the consequence Practical support for dementia patient through the consequence 'Meal preparation'
Please choose a want and its corresponding group of consequences 4
You chose the want: Diningtable with the consequences ...
For the selected functional consequence, the quality consequences - per scale - are:
We found the following scale of quality consequences: Diet
Please assign a score from 1 (not important) to 10 (vital) to each of these consequences ...

Table 2
Sample of the tool output for the meal preparation scenario.

Case. We assume here again our first scenario: an informal carer that searches for Meal preparation services.

The informal carer selects 'Cannot cope anymore, what can help?'. Next, this need is satisfied by four functional consequences (see Fig. 3), from which the customer selects 'Practical support for person with dementia'. 'Practical support for person with dementia', in turn, is further specified further by, amongst others, the consequences 'Possibility to loan e.g. an (electrical) wheelchair' and 'Meal preparation'.

Subsequently, the reasoning process finds that these consequences must be leaf consequences because they no longer 'consist of' other, more detailed, consequences. The reasoning process therefore proceeds to find the wants to which these leaf consequences belong. For example: The consequence 'Meal preparation' is part of the wants 'Dinnerdelivery' and 'Diningtable', while the consequence 'Possibility to loan e.g. an (electrical) wheelchair' is part of the want 'Loaningservice' (see Fig. 3).

The reasoning process now uses the found wants as a bootstrap mechanism. For example: Contained in the want 'Diningtable', it finds the additional functional consequences 'Social contacts informal carer' and 'Social contacts for person with dementia'.

The tool output in table 2 shows how the customer would be presented with the consequences from the want Dining table.

Let us assume that an informal carer selects the want 'Dining table', because besides 'Meal preparation', it also offers the consequences 'Social contacts for person with dementia' and 'Social contacts for informal carer'.

Outcome of this step: The set of functional consequences {'Meal preparation', 'Social contacts informal carer', 'Social contacts person with dementia'}.

6.3.2. Choose Additional Consequences

Reasoning steps. For a selected functional consequence, we derive its quality consequences by following the relation 'qualifies' in our customer perspective ontology. These quality consequences are then grouped by scale by (1) deriving, for each quality consequence, its scale by following the relation 'provided by' between quality consequence and scale and (2) grouping together quality consequences that are defined on the same scale.

Now that the consequences are grouped per scale, we let the customer decide, per scale, on consequence prioritization. Depending on the *type* of scale (recall that different types of scales exist, i.e.: Nominal or ordinal) the customer is presented with two different prioritization tasks:

- When consequences are defined on a *nominal* scale, the customer is asked to assign to each of the consequences an importance value ranging from 1 (unimportant) to 10 (must-have, i.e. the offered service bundle should always include this consequence);
- When consequences are defined on an *ordinal* scale, the customer is asked to assign a preference ordering to the *scale only* and not to the consequences in the scale. For the scoring of consequences, we use the ranking from best to worst that is inherent to an ordinal scale. How we convert this best-to-worst ranking of consequences into a score, is discussed in the next step (compose and rank service bundles, see below).

Finally, by default, the selected functional consequences, as well the default quality consequences of a want, receive an importance ranking of '10' (must have). We can now use the customer preferences expressed in terms of their consequences to compose the appropriate service bundles.

Case. For the functional consequence 'Meal preparation', the informal carer indicates her dietary needs by scoring the quality consequences from the nominal scale 'Diet'. We assume that our informal carer scores 'Fish as main course' with 8 and 'Flesh as main course' with 6. In addition, s/he scores the quality consequence 'Kosher' as 1 and the quality consequence 'Sugar free' as 2.

Next, the reasoning process finds that the want 'Diningtable' contains the default quality consequences 'In person' and 'Warm' (see also Fig. 3) and attaches a preference score of 10 to indicate that these are must-haves. Note that these default consequences are enforced by the idea behind a dining table: Persons coming together in a care home to eat a hot meal.

Finally the chosen functional consequences 'Meal preparation', 'Social contacts for person with dementia' and 'Social contacts for informal carer' will be given an importance score of 10 by default to indicate that these are must-have features.

Outcome of this step: The set of functional and quality consequences and their respective importance scores {Meal preparation (10), Social contacts informal carer (10), Social contacts person with dementia (10), Warm (10), In person(10), Kosher (1), Sugar free (2), Flesh as a main course (6), Fish as main course (8)}.

6.3.3. Find service bundles

Reasoning steps. We first match the set of consequences desired by the customer to supply-side consequences. We can do this because the concept 'consequence' provides the bridging connection between the customer and supplier perspective on services. The computational result of this consequence matching is the subset of all supply-side consequences that, together with the prioritization scores provided by the customer, can be used to reason about (1) finding (composing) service bundles and (2) ranking service bundles according to prioritization scores.

We first search service bundles that can satisfy all 'must-have' consequences. To find these bundles, we search for each must-have consequence supplier-specific resources that jointly provide the consequence. We then find bundles containing these supplier-specific resources. Recall that, as discussed in the supplier-perspective ontology, resources can also realize consequences.

The result is a set of bundles per each *individual* must have-consequence. The set of bundles satisfying *all* must-have consequences then is an intersection of the set of bundles generated for the must-haves. So, if must-have consequence A is satisfied by the set of bundles {X, Y, Z} and must-have consequence B is satisfied by the set of bundles {W,X}, the set of bundles containing both must-have consequences A and B is {X}.

Case. The matching of customer side and supply side consequences results in the following subset of *supply-side* functional and quality consequences and their respective importance scores: {social contacts for person with dementia (10), social contacts informal carer (10), meal preparation (10), sugar free (2), kosher (1)}. For reference, see the supplier perspective service catalogue in Fig. 7.

The reasoning process now finds the pregenerated service bundles that can satisfy all 'must-have' consequences:

1. The reasoning process finds the set of bundles {'Dining table' (Torendael), 'Dining table' (de Buitenhof)} for the must-haves 'social contacts informal carer' and 'social contacts for person for with dementia'. Both bundles realize both consequences through the service property 'in group, with other persons with dementia and their informal carers' of the supplier-specific 'Meal' resources.
2. The tool finds the set of bundles {'Dining table' (Torendael), 'Dining table' (de Buitenhof), 'Dinner delivery' (Schinkelshaven)} for the consequence 'Meal preparation', because all these bundles can satisfy this consequence through the resource 'Meal' (see Fig. 7 for reference).

When the reasoning process now intersects above sets, it finds the bundles {'Dining table' (de Buitenhof)}, {'Dining table kosher'(Torendael)}, and {'Dining table' (de Buitenhof), Transportation (Stadsmobiel BV)}.
Outcome of this step: The set of supplier-specific service bundles {'Dining table' (de Buitenhof)}, {'Dining table kosher' (Torendael)}.

6.3.4. Rank service bundles
Reasoning steps. For each bundle, we calculate a ranking score by employing the following multi-attribute scoring formula:

$$SB_i = \sum_{j=1}^{n} \frac{w_j}{10} v_{ij} \tag{1}$$

Here, SB_i is the ranking score for service bundle i, w_j is the importance score of consequence j as provided by the customer and v_{ij} is the numerical value for the consequence j of service bundle i, that indicates if a consequence is present in a bundle. For an illustration, see the case example.

Observe that two factors are important for calculating the importance score of a service bundle: (1) the importance score of a consequence as provided by the customer, which is known, and (2) the numerical value of a consequence that indicates its presence in a bundle. The numerical value of a consequence is calculated based upon the scale that it belongs to:

- We score consequences that belong to a *nominal* scale in a binary way; if present in a bundle, a consequence scores 1, otherwise 0.
- When a consequence belongs to an *ordinal* scale we convert the position in the *qualitative* best-to-worst ranking of the consequence to a *quantitative* ranking score that we can use in a ranking calculation. For this conversion, we use the Rank-Order Centroid method (ROC, Hutton Barron and Barrett (1996)) which can transform a qualitative ranking of a consequence into a quantitative ranking whose values are normalized to a value [0...1]. ROC is a well-established tool used in management literature to support decision-making.
  ROC uses the following formula to convert qualitative rankings to quantitative ones:

$$C_k = \frac{\sum_{j=k}^{N} \frac{1}{j}}{N} \tag{2}$$

  Where C_k is the ROC score for a consequence at the k-th place in a qualitative ranking of consequences and N the total number consequences ranked.

Case. We calculate a ranking score for the found bundles based upon consequences with an importance score below 10 (of course, all found bundles score the same on must-have consequences). In this scenario, the tool thus calculates an importance score for each bundle using as input the consequences 'Sugar free', 'Kosher', 'Fish as main course' and 'Flesh as main course'.

To explain multi-attribute scoring, consider the consequence 'Kosher' from the bundle {Diningtable (Torendael)}. The consequence 'Kosher' would score (1 * 0.1), because (1) it receives the score '1' by being a consequence from a nominal scale. Recall that consequences from nominal scales are calculated

<Match of the chosen consequences to the supply side consequences>
<Tool traces consequences to service bundles that realize these consequences>
1: The service bundle Diningtable Torendael can satisfy all desired consequences. Please find below what this bundle can provide you with
*Social contacts for informal carer, Social contacts for person with dementia, Meal preparation
Additionally, this bundle contains the following quality consequences:
* Diet: Kosher for the functional consequence Meal preparation
* Contact medium: in person for the functional consequence Social contacts for informal carer and Social contacts for person with dementia
* Preparation: Warm for the functional consequence Meal preparation
Score bundle = (0.1*1.0) for Kosher
To acquire the bundle : Diningtable Torendael kosher you have to give up the following:
*Fee Dining table Torendael kosher, Arrange own transportation
For this bundle, we found the following pricing model:
*Pricing model Diningtable Torendael kosher. This pricing model is of the type: single discount pricing model.
The price of this bundle is build up as follows:
* The usage-based pricing model Pricing model Diningtable Torendael kosher is build up as follows: 7 euro for each time. This usage-based pricing model is for the individual service Diningtable Torendael kosher
2: The service bundle Diningtable Welzijn ZuiderAmstel can satisfy all desired consequences. Please find below what this bundle can provide you with
....
So the possible bundles, sorted according to preference, are:
1: Diningtable de Buitenhof with the score: 1.0
2: Diningtable Torendael kosher with the score: 0.1
Would you also like to score all negative consequences to see how a bundle scores on a trade-off between benefits and sacrifices?(y/n)

Table 3

Sample of the tool output for the meal preparation scenario, step 'find service bundles'.

in a binary way: 1 or 0. and (2) 'Kosher' receives the score 0.1 from the informal carer (its importance score divided by 10, cf. multi-attribute formula 2).

Now, using multi-attribute scoring, we arrive at the following scores:

- 1.0 for 'Dining table' (de Buitenhof) ((0.1 * 1) for the consequence flesh as main course + (0.7* 1.0) for the consequence fish as main course)
- 0.3 for 'Dining table kosher' (Torendael) ((0.1 * 1) for the consequence kosher).

.

Outcome of this step: A set supplier-specific service bundles, ranked according to how well they fit with the preferences of the informal carer (see table 3).

6.3.5. Trade-off feature/opportunity cost

Find full set of consequences. Next we find, for each proposed bundle, additional consequences that a customer must also acquire. Importantly, these additional consequences mainly represent objects of value that a customer has to give up to acquire a service bundle, such as a fee or customer lock-in.

To find additional consequences we find, for each service bundle, additional service ports found by reviewing the bundle's service interface. Taking these additional service ports as a starting point we then derive consequences by (1) finding the resources attached to the service ports, (2) deriving the service properties from these resources and finally (3) finding the consequences attached to these service properties and/or resources.

Case. Consider for example the service bundle {Dining table (Torendael)}. From the service interface of this bundle, we find the additional consequences 'Dining table fee' and 'Arrange own transportation'.
Outcome of this step: Service bundles and their full set of consequences. For example: The bundle {Dining table (Torendael)} provides set of consequences {'Meal preparation', 'Social contacts person with dementia', 'Social contacts informal carer'} supplemented with the set of consequences {'Dining table fee', 'Arrange own transportation'}.

Find pricing model and present for each bundle its full set of consequences. We present the found bundles in a ranking to the customer, where the ranking is based upon the multi-attribute score discussed in

Sect. 6.3.4. In addition, for each bundle, the customer is presented with a specification of the consequences s/he receives from a bundle and the consequences s/he has to give up to acquire a bundle. Finally, the customer is presented with a specification of monetary objects by means of a *pricing model* (see section 5.3).

This pricing model of a bundle, the reasoning process finds by following the '*Has*' relation of the service interface of that bundle (see the pricing model ontology, Fig. 6).

Case. Consider again the bundle {'Dining table' (Torendael)}. For this bundle, we find the single discount pricing model 'Pricing model dining table Torendael'. This pricing model consists of the single usage based pricing model of the Dining table Torendael service: 7 euros for each time, with a discount of 0%.

6.3.6. Score negative consequences.

The customer can now choose to also score *negatively valued* consequences. If selected, the customer is again asked to attach a score from 1 to 10 to each consequence. However, as opposed to scoring positive consequences, '1' is now used to indicate that the presence of a consequence in a service bundle is not negatively valued, while '10' is used to indicate that a consequence should be *absolutely absent* from a bundle.

Based upon the scores attached to the negatively valued consequences, the reasoning process then again uses the scoring mechanism discussed in Sect. 6.3.4 only now with the difference that:

1. Each bundle that satisfies a negatively valued consequence with a score of 10 is automatically discarded, irrespective of other consequences satisfied by that bundle.
2. We compute again a score for each bundle remaining after (a) by using a multi-attribute scoring formula, only now by *subtracting* the scores of the negatively valued consequences from the bundle scores.

Note here that if a bundle is rejected, the critique step as already explained is invoked to find the cause for rejection.

The customer now has the option to either select a bundle from the ranking or, in case s/he finds the costs incurred for the bundles too high, to go back to the step 'choose consequences' and modify her consequences.

Case. The customer is presented with the ranked bundles:

1. Dining table (de Buitenhof). Score 1.0
2. Dining table (Torendael) . Score: 0.3

For each bundle, the informal carer is presented with (1) the positively valued consequences, (2) the negatively valued consequences, and (3) the pricing model. For an example specification of the bundle 'Dining table'(Torendael), see table 3.

Next, the informal carer also scores negatively valued consequences. We assume s/he is content with the dining table fees from both {Dining table (Torendael)} and {Dining table (de Buitenhof)}, but attaches 10 to 'Arrange own transportation'.

The reasoning process invokes the critique step for the consequence 'Arrange own transportation'.It finds that both dining table services contain the consequence 'Arrange own transportation' (for reference, see Fig. 9) and informs the informal carer that acquiring the set of positive consequences from these services, i.e. {'Meal preparation', 'Social contacts for informal carer', 'Social contacts for person with dementia'}, implies having to arrange one's own transportation. Finally, both diningtable services are discarded.

Outcome of this step: No possible bundles

6.3.7. Choose consequences, find and rank service bundles

Reasoning steps. To modify her consequences, the customer can now return to the point in the consequence ladder of the customer perspective catalog that requires modification. So, if a customer finds that a set of consequences {X,Y,Z} as implied by a want always implies a won't have consequence, then the customer chooses to return to a point in the consequence ladder prior to selecting the want that implies consequences {X,Y,Z}.

Following consequence modification, the reasoning process for finding service bundles is similar to the reasoning process we described thus far, and which is depicted in Fig. 8. So: we again follow the steps 'Choose consequences', 'Find service bundles', 'Trade-off positive/negative consequences', only then for the modified consequences. We illustrate this in the case example following this paragraph.

Case. The informal carer can now return to any point in the consequence ladder to adjust consequences.

From the previous step, the informal carer has learned that the set consequences belonging to the want 'Diningtable' ({'Meal preparation', 'Social contacts for informal carer', 'Social contacts for person with dementia'}) requires arranging one's own transportation. Therefore the informal carer chooses to return to a point in the consequence ladder prior to selecting the want Diningtable: s/he selects the consequence 'Practical support for person with dementia' (for reference, see the customer perspective catalog Fig. 3).

Thereafter, the informal carer finds that the want 'Dinner delivery' also contains the consequence 'Meal preparation' - the consequence of interest - but without the Diningtable consequences that imply a social function.

As before, the customer now prioritizes dietary needs and preparation options through scoring quality consequences for the functional consequence 'Meal preparation'.

Thereafter, the reasoning process now finds and ranks services again and presents the following ranked service bundles to the informal carer:

- Dinner delivery Schinkelshaven: 0.9
- Dinner delivery Amsterdam home care: 0.8

We assume that the informal carer is satisfied with the small fee that has to be paid for each service bundle, and that s/he decides to acquire the bundle 'Dinner delivery' (Schinkelshaven).

Outcome of this step: The informal carer selects the bundle 'Dinner delivery' (Schinkelshaven).

6.3.8. Review value-enhancing consequences

To illustrate how we reason about value-enhancing consequences, we use our second scenario: Loaning a wheelchair and finding related, value-enhancing services. To focus our discussion on value-enhancing consequences, we assume that the customer has already selected, using the reasoning process discussed thus far, the basic bundle: {'Loaningservice' (Amsterdam homecare)}.

Find value-enhancing consequences After the customer selects a bundle, we review if there are any value-enhancing consequences (either core/enhancing or optional bundled consequences) for that bundle. Recall that in our generic reasoning process (Fig. 8) this step is indicated by the guard [value-enhancing consequences].

To find value-enhancing consequences, the reasoning process uses three steps: (1) finding the wants for the service elements of the selected bundle. Since the concept of a want also exists on the supply side (see the customer perspective ontology in Fig. 4) we can classify service elements as wants. (2) matching the supplier perspective wants to the customer perspective wants, and finally (3) reviewing, on the customer perspective, if there exist any value-enhancing wants for the matched want.

Subsequently, if there exist value-enhancing wants, we first derive the core consequence from the core want, and review what value-enhancing consequence from the value-enhancing want provides the reason for this value-addition.

Case. First, we find wants for the service elements of the selected bundle {Loan equipment (Amsterdam homecare)}. The reasoning process finds that the supplier-specific service element 'Loan equipment' from

Possibility to loan e.g. an (electrical) wheelchair
Additionally, this bundle contains the following quality consequences:
* Duration: => 6 months for the functional consequence Possibility to loan e.g. an (electrical) wheelchair
Score bundle = (0.9*1.0)
To acquire the bundle : Loaningservice Amsterdam homecare you have to give up the following:
* Arrange own transportation, AWBZ indication

Table 4

A loaningservice that provides the consequence 'Loaning of (e.g. electrical) wheelchair'.

this bundle is classified as the want 'Loaningservice'. As stated, we can find this classification because the supplier ontology (see Fig. 4) allows for classifying a service element as a want.

Second, we match the supplier perspective want 'Loaningservice' to the customer perspective want 'Loaningservice'.

Third, the want 'Loaningservice' is used in the customer perspective catalog (see Fig. 3) to find the value-enhancing wants 'Transportation' and 'Handyman'. Next, for the core consequence 'Possibility to loan e.g. an (electrical) wheelchair' from the want 'Loaningservice', we find the value-enhancing consequences 'Transportation' and 'Adjustments to home'. Finally, these value-enhancing consequences are found to belong the value-enhancing wants 'Transportation' and 'Handyman' respectively.

Outcome of this step: The informal carer is now presented with the following options (table 5):

For the bundle Loaningservice Amsterdam homecare we found the following optionally bundled wants:
*The want: Transportation through the consequences: Transportation
Are you interested in this optional bundling want? (y/n) n
*The want: Handyman through the consequences: Adjustments to home
Are you interested in this optional bundling want? (y/n)

Table 5

Eliciting value-enhancing consequences

Find service bundles for value-enhancing consequences As in the step 'choose consequence', the customer is now asked to make a choice based on a combination of a want (the solution) and a consequence (why the solution is valuable) only now for a value-enhancing service. If the customer is interested in the value-enhancing want, we iterate again through the steps described so far to derive a set of possible service bundles (i.e.: the steps 'choose consequences', 'find and rank service bundles' and trade-off feature/opportunity costs'), only now for the value-enhancing wants.

Case. We assume that the informal carer chooses the functional consequence 'Adjustments to home' from the want 'Handyman'. Next, the informal carer is requested to score the quality consequences 'Minor' and 'Large' for 'Adjustments to home', and scores both consequences as '2'.

The tool now supplements these chosen value-enhancing consequences with those from the bundle 'Loaning service homecare'. This results in the following input for the reasoning process: {Adjustments to home(10), Minor(2), Large(2), Possibility to loan e.g. an (electrical) wheelchair(10), longer than 6 months(9), shorter than 6 months(4)}.

Since 'Adjustments to home' is a must-have consequence, the reasoning process finds only bundles that can satisfy this consequence: {Loan equipment (Amsterdam hc), Arrange Handyman (Cordaan)} (For reference, see the supply-side catalog in Fig. 7).

Outcome of this step (see table 6):

7. Tool support

To ensure the computational adequacy of the e^3*service* reasoning process, we implemented its reasoning steps in a software reasoner. We create ontology instantiations with the ontology editor Protege [3]. Protege

[3] http://protege.stanford.edu/

3: The service bundle {Loan equipment (Amsterdam hc), Arrange handyman (Cordaan)} can satisfy all desired consequences. Please find below what this bundle can provide you with
*Adjustments to home, Possibility to loan e.g. an (electrical) wheelchair
Additionally, this bundle contains the following quality consequences: …

Table 6
Sample of tool output: matched service bundles

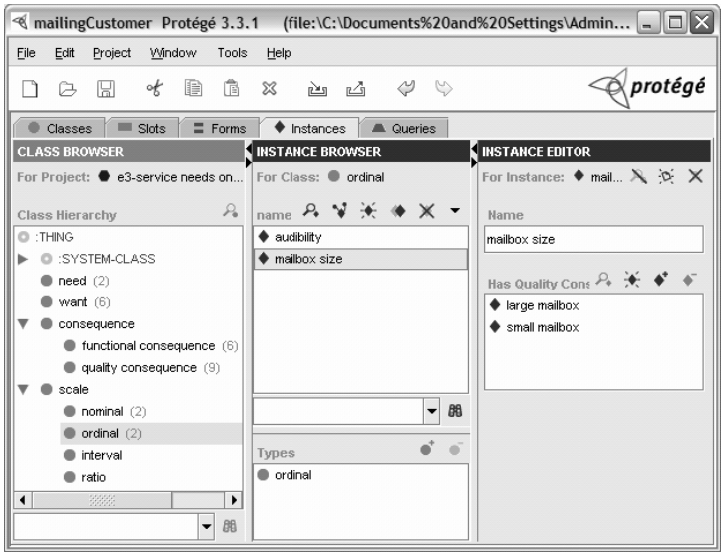

Fig. 12. Instantiation of the e^3*service* customer perspective service catalogue in Protege

provides an environment for creating and editing an ontology and ontology instantiations, and exporting these to various formats, including XML-based formats such as RDF/S and OWL. For an example, see Fig. 12. Here, for a representation of a customer catalogue of ISP-services in Protege, we see the ordinal scale 'mailbox size' with two quality consequences: 'small mailbox' and 'large mailbox'.

We export ontology instantiations made with Protege to RDF-format and use these exported instantiations as input for our software reasoner.

Our software reasoner, in turn, uses a RDF-parser from the Jena-environment[4] to interpret RDF-files and reason with these.

The software reasoner follows the inferencing steps from e^3*service* over the RDF-representations of the ontology instantiations. It follows exactly the steps detailed in the previous section: No steps are added, modified or removed.

8. Conclusions and future work

8.1. Key points

In this paper we have proposed e^3*service* , a multi-perspective ontology to represent needs, and multi-supplier service bundles. With e^3*service* we introduced a formal theory of consequences. We showed that consequences, from the well-established marketing theory means-ends chaining, are a central concept for bridging between (customer perspective) needs and (supplier perspective) services. Also, we showed how to use consequences for reasoning about translating customer needs into suitable service bundles.

We relied on ontologies to create a formal theory of consequences. Ontologies provide a common vocabulary that enterprises can use for expressing their service knowledge. As such, ontologies seem

[4]Jena is a java-library that supports the development of semantic web applications. See: http://jena.sourceforge.net/

a good first step towards generating bundles of services in a multi-supplier setting. For example: By using *e*^{3}*service* , enterprises agree on expressing services through their valuable outcomes, instead of the technical properties of these services.

Furthermore we introduced PCM^2, a reasoning method for automating the needs-driven bundling process. We discussed how PCM^2 is inspired by the PCM family of problem solving methods, but whereas PCM-based methods focus on modifying solutions only, PCM^2 modifies both requirements *and* solutions. This is because, as the reasoning on service bundling progresses, new requirements are considered, requirements are prioritized differently and requirements may be removed altogether. As a result, we consider the requirements space as a first class citizen. For *e*^{3}*service* the modification of requirements and solutions is equally important.

8.2. Further research

The formal theory of consequences introduced in this paper provides a next step towards fulfilling our envisioned research goal: To generate bundles of commercial ICT services (semi-) automatically, based upon a (complex) customer need. However, realizing this research goal obviously requires more than introducing an ontology. We therefore suggest the following research directions.

Reasoning about bundling services from a supplier perspective In our work, we concentrate on the customer perspective on services. We assume the supply-side service bundles to be generated upfront, using the service dependencies defined by Akkermans et al. (2004). However, closer inspection reveals that these supply side dependencies still have several shortcomings that need to be addressed before they are fit for dynamically bundling services in a multi-supplier setting. First, service dependencies always exist between supplier-specific services, which makes it very difficult to model considerations such as 'We need an internet connection for this VoIP service, and do not care who provides this as long as the internet connection satisfies the following minimum requirements...'. We feel that the focus should shift from representing supplier-specific services to representing the minimum requirements that a service should adhere to. Second, reasoning on price bundling (receiving a discount on the bundle price) and product bundling (receiving positive consequences additional to the sum of the consequences of the individual services in the bundle) should also be part of the effort to generate service bundles dynamically. These considerations occur too often in practice to be ignored. Consider for example our healthcare case, where the customer receives the additional positive consequence 'meeting the same personnel' when s/he uses the services 'recreational activities' and 'meeting a physician' from the same care centre.

Introduce constraints on the customer perspective also In this paper, we use positive consequences as input for the reasoning process. Only after matching, the customer receives feedback on the constraints of service bundles. To further experiment with a PCM-type of PSM, we may introduce constraints on the supply side also. A good starting point is the micro-economic concept *maximum-willingness-to-pay*. Concerning variations on the current reasoning process, one can for example link the concept of willingness-to-pay to a customer need. A customer can then state his problem and his constraints, and gradually narrows down the found service bundles using consequences as additional requirements.

Business models for realizing bundling in a multi-supplier setting We -as researchers- have modelled the services and needs for this paper, thereby assuming the role of an intermediary. In practise however, an intermediary model implies that there is one party that controls how services are modelled, and that decides what suppliers can participate in the bundling process and what suppliers cannot. This concentration of power, residing at one intermediary, seems undesirable.

As a direct opposite of an intermediary business model, one may consider a fully decentralized environment. Herein, service bundles are found in a peer-to-peer fashion. Each supplier locally expresses and maintains its service knowledge, by using a service ontology for example, whereupon the network generates service bundles. Given that suppliers can increase sales by advertising their services to the network, it seems that there are reasonable incentives for participation. However, we also foresee difficulties with a fully decentralized environment. For one, it introduces inconsistencies between service descriptions from

different providers, because each supplier would model the valuable outcomes of their services *locally*. For example, different suppliers may refer to an e-mailing capability in different ways, such as 'e-mailing' 'send and receive e-mail' and 'e-mailing services'. For bundling in a decentralized environment, such inconsistencies need to be addressed. To address the limitations of centralized and decentralized environments, we suggest to explore a mix between the two. For one, in a mixed environment trusted third parties can be used to maintain a consistency in the service knowledge expressed by different suppliers.

Finally, all scenarios - centralized, de-centralized and a mixed between these - need to cope with the limited openness of providers. It is our experience that providers will happily expose all advantages of their services, but that they are hesitant when it comes to expressing service limitations such as customer lock-in and payment. Here, a possible starting point may be to emphasize that openness is also a form of competitive advantage.

References

Akkermans, H., Baida, Z., Gordijn, J., Pena, N., Altuna, A., and Laresgoiti, I. (2004). Value webs: Using ontologies to bundle real-world services. *IEEE Intelligent Systems*, 19(4):57–66.

Ambler, C. (1998). Developing a Product Classification System for the United States. *US Census Bureau*.

Arndt, J. (1978). How broad should the marketing concept be? *Journal of Marketing*, 42(1):101–103.

Aschmoneit, P. and Heitmann, M. (2002). Customer centred community application design: Introduction of the means-end chain framework for product design of community applications. *International Journal on Media Management*, 4(1):13–20.

Chandrasekaran, B. (1990). Design problem solving: A task analysis. *AI magazine*, 11(4):59–71.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1. URL:http://www.w3.org/TR/wsdl, last accessed on December 31, 2012.

de Kinderen, S. (2010). *Needs-driven service bundling in a multi-supplier setting - the computational e^3service approach*. PhD thesis, VU University Amsterdam.

de Kinderen, S., Gordijn, J., and Akkermans, H. (2009). Reasoning about customer needs in multi-supplier ict service bundles using decision models. In *Proceedings of the 11th International Conference on Enterprise Information Systems*, pages 131–136.

de Kinderen, S., Gordijn, J., Dröes, R., and Meiland, F. (2009). A computational approach towards eliciting needs-driven bundles of healthcare services. In *Proceedings of the 22nd Annual Bled Conference*, Maribor, SL.

De Leenheer, P., Cardoso, J., and Pedrinaci, C. (2013). Ontological representation and governance of business semantics in compliant service networks. In *Proc. of the International Conference on Exploring Service Sciences*, number 143 in LNBIP, pages 155–169. Springer-Verlag, Heidelberg.

Dröes, R., Meiland, F., Doruff, C., Varodi, I., Akkermans, H., Baida, Z., Faber, E., Haaker, T., Moelaert, F., Kartseva, V., et al. (2005). A dynamic interactive social chart in dementia care. *Medical and Care Compunetics*, 2:210–224.

Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall.

Fensel, D., Lausen, H., Polleres, A., de Bruin, J., Sollberg, M., Roman, D., and Domingue, J. (2006). *The Web Service Modeling Ontology*. Springer-Verlag, Heidelberg, Germany.

Ferrario, R. and Guarino, N. (2012). Commitment-based modeling of service systems. In Snene, M., editor, *Exploring Services Science*, volume 103 of *Lecture Notes in Business Information Processing*, pages 170–185. Springer-Verlag, Heidelberg.

García, F., Bertoa, M., Calero, C., Vallecillo, A., Ruíz, F., Piattini, M., and Genero, M. (2006). Towards a consistent terminology for software measurement. *Information and Software Technology*, 48(8):631–644.

Grimnes, M. and Aamodt, A. (1996). A two layer case-based reasoning architecture for medical image understanding. In *Advances in Case-Based Reasoning*, pages 164–178. Springer-Verlag, Heidelberg.

Grönroos, C. (2007). *Service management and marketing: customer management in service competition*. Wiley India Pvt. Ltd.

Guiltinan, J. (1987). The price bundling of services: a normative framework. *Journal of Marketing*, 51(2):74–85.

Gutman, J. (1997). Means–end chains as goal hierarchies. *Psychology and Marketing*, 14(6):545–560.

Gutman, J. and Reynolds, T. (1982). A means-end chain model based on consumer categorization processes. *Journal of Marketing*, 46(2):60–72.

Gutman, J. and Reynolds, T. (1988). Laddering theory-analysis and interpretation. *Journal of Advertising Research*, 28(1):11–31.

Herrmann, A., Huber, F., and Braunstein, C. (2000). Market-driven product and service design: Bridging the gap between customer needs, quality management, and customer satisfaction. *International Journal of production economics*, 66(1):77–96.

Holbrook, M. (1999). *Consumer value; a framework for analysis and research*. Routledge, 29 West 35th Street, New York, NY 10001.

Hutton Barron, F. and Barrett, B. (1996). Decision quality using ranked attribute weights. *Management Science*, 42(11):1515–1523.

Karpen, I., Bove, L., and Lukas, B. (2012). Linking service-dominant logic and strategic business practice: A conceptual model of a service-dominant orientation. *Journal of Service Research*, 15(1):21–38.

Kinderen, de, S. and Gordijn, J. (2008). Reasoning about substitute choices and preference ordering in e-services. In Bellahsène, Z. and Léonard, M., editors, *Proc. of CaISE*, pages 390–404.

Kotler, P. (2000). *Marketing Management*. Prentice Hall, Upper Saddle River, NJ.

Loudon, D. and Della-Bitta, A. (1993). *Consumer behaviour*. McGraw-Hill, New York.

Maglio, P., Srinivasan, S., Kreulen, J., and Spohrer, J. (2006). Service systems, service scientists, SSME, and innovation. *Commun. ACM*, 49(7):81–85.

McCarthy, W. (1982). The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *Accounting Review*, 57(3):554–578.

Mentzer, J., Rutner, S., and Matsuno, K. (1997). Application of the means-end value hierarchy model to understanding logistics service value. *International Journal of Physical Distribution & Logistics Management*, 27(9/10):630–643.

Michael, L., Johnson, D., and Renaghan, L. (1999). Adapting the qfd approach to extended service transactions. *Production and Operations Management*, 8(3):301–317.

Miranda, de, B., Baida, Z., and Gordijn, J. (2006). Modeling pricing for configuring e-service bundles. In Walden, P., Markus, M. L., Gricar, J., and Lenart, G., editors, *Proceedings of the 19th BLED conference (eValues)*, Maribor, SL.

Monroe, K. (1990). *Pricing: Making profitable decisions*. McGraw-Hill Companies.

Motta, E. (1999). Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving.

Motta, E., Stutt, A., Zdrahal, Z., O'Hara, K., and Shadbolt, N. (1996). Solving VT in VITAL: a study in model construction and knowledge reuse. *International Journal of Human-Computers Studies*, 44(3):333–371.

Normann, R. (2000). *Service Management - strategy and leadership in service business*. Wiley. Third edition.

Normann, R.; Ramirez, R. (1993). From value chain to value constellation: Designing interactive strategy. *Harvard Business Review*, 71:65–77.

Osterwalder, A. (2004). *The Business Model Ontology - a proposition in a design science approach*. PhD thesis, University of Lausanne, HEC.

Poeck, K., Fensel, D., Landes, D., and Angele, J. (1996). Combining KARL and CRLM for designing vertical transportation systems. *International Journal of Human-Computers Studies*, 44(3-4):435–467.

Poels, G. (2010). The resource-service-system model for service science. In *ER Workshops*, pages 117–126. Springer-Verlag, Heidelberg.

Polanyi, M. (1966). *The Tacit Dimension*. Anchor Books.

Quinn, J. and Paquette, P. (1990). Technology in services: creating organizational revolutions. *Sloan Management Review*, 31(2):67–78.

Razo-Zapata, I., De Leenheer, P., Gordijn, J., and Akkermans, H. (2011). *Handbook of Service Description: USDL and its Methods*, chapter Service Network Approaches, pages 45 – 74. Springer-Verlag, Heidelberg.

Razo-Zapata, I., De Leenheer, P., Gordijn, J., and Akkermans, H. (2012). Fuzzy verification of service value networks. In Ralyte, J. and Franch, X., editors, *Proc. of CAiSE*, pages 95–110. Springer-Verlag, Heidelberg.

Schreiber, A. and Terpstra, P. (1996). Sisyphus-VT: a CommonKADS solution. *International Journal of Human-Computer Studies*, 44(3-4):373–402.

Schreiber, A. T., Akkermans, J., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B. (2000). *Knowledge engineering and management: The CommonKADS methodology*. MIT press.

Solomon, M. (2003). *Consumer behavior: buying, having and being*. Prentice Hall, Upper Saddle River, New Jersey 07458.

Stremersch, S. and Tellis, G. (2002). Strategic bundling of products and prices: A new synthesis for marketing. *Journal of Marketing*, 66(1):55–72.

Tapscott, D., Ticoll, D., and Lowy, A. (2000). *Digital Capital - Harnessing the Power of Business Webs*. Nicholas Brealy Publishing, London, UK.

ten Teije, A. and van Harmelen, F.and Wielinga, B. (2004). Configuration of web services as parametric design. In Motta, E., Shadbolt, N., Stutt, A., and Gibbins, N., editors, *Proceedings of the 14th International EKAW Conference*, number 3257 in Lecture Notes in Artificial Intelligence, pages 321–336, Whittleburry Hall, UK. Springer-Verlag, Heidelberg.

ten Teije, A., van Harmelen, F., Schreiber, A., and Wielinga, B. (1998). Construction of problem-solving methods as parametric design. *International Journal of Human-Computer Studies*, 49(4):363–389.

Thomas, O. and Fellmann, M. (2007). Semantic EPC: Enhancing process modeling using ontology languages. In *SBPM*.

Tung, W., Capella, L., and Tat, P. (1997). Service pricing: A multi-step synthetic approach. *Journal of Services Marketing*, 11(1):53–65.

van der Raadt, B., Schouten, S., and van Vliet, H. (2008). Stakeholder perception of enterprise architecture. In Morrison, R., Balasubramaniam, D., and Falkner, K., editors, *Software Architecture*, volume 5292 of *Lecture Notes in Computer Science*, pages 19–34. Springer-Verlag, Heidelberg.

Vargo, S. and Lusch, R. (2004). Evolving to a new dominant logic for marketing. *Journal of Marketing*, 68(1):1–17.

Vargo, S. and Lusch, R. (2008). Service-dominant logic: continuing the evolution. *Journal of the Academy of Marketing Science*, 36(1):1–10.

Woodruff, R. (1997). Customer value: The next source for competitive advantage. *Journal of the Academy of Marketing Science*, 25(2):139–153.

Zeithaml, V., Parasuraman, A., and Berry, L. (1985). Problems and strategies in services marketing. *The Journal of Marketing*, 49(2):33–46.