Fuzzy Verification of Service Value Networks

Iván S. Razo-Zapata[1], Pieter De Leenheer[1,2],
Jaap Gordijn[1], and Hans Akkermans[1]

[1] VU University, Amsterdam, The Netherlands
{i.s.razozapata,p.g.m.de.leenheer,j.gordijn,elly.lammers}@vu.nl
[2] Collibra nv/sa, Brussels, Belgium

Abstract. Service Value Networks (*SVNs*) represent a flexible design for service suppliers to offer attractive value propositions to final customers. Furthermore, networked services can satisfy more complex customer needs than single services acting on their own. Although, *SVNs* can cover complex needs, there is usually a mismatch between what the *SVNs* offer and what the customer needs. We present a framework to achieve *SVN* composition by means of the propose-critique-modify (PCM) problem-solving method and a Fuzzy Inference System (FIS). Whereas the PCM method composes alternative *SVNs* given some customer need, the FIS verifies the fitness of the composed *SVNs* for the given need. Our framework offers not only an interactive dialogue in which the customer can refine the composed *SVNs* but also visualizes the final composition, by making use of e^3-*value* models. Finally, the applicability of our approach is shown by means of a case study in the educational service sector.

Keywords: Service Value Networks, PCM, Composition, Verification, Fuzzy Logic.

1 Introduction

Commercial services do not only cover an important segment of countries' economies but also are thought to play a big role in the Future Internet [18,17]. Since networked services can cover more complex customer needs, to offer service-based solutions for customers in a changing environment like the Web, service providers must be able to interact with other business entities to create *service networks (SNs)* [18]. In general, since commercial services are economic activities offering tangible and intangible value resources, value aspects within these networks must be also analyzed [2]. More specifically, it is important to understand how customers and suppliers can co-create *service value networks (SVNs)*.

We have previously implemented part of our framework describing how SVNs can be (semi)automatically composed to meet specific (complex) customer requirements by means of networking offerings coming from different service suppliers [14,19,21]. In this paper, our contribution is twofold. First, we present a problem-solving method to allow composition of *SVNs* based on customers' and suppliers' perspectives. Customers try to find answers to their needs whereas

J. Ralyté et al. (Eds.): CAiSE 2012, LNCS 7328, pp. 95–110, 2012.

suppliers can be organized into a *SVN* to satisfy the customer needs. Moreover, in case none of the composed *SVN* offers a good answer to the customer, the framework allows the customer to give feedback for the composed *SVNs* and restart the composition process until (s)he finds a *SVN* that fits her/his need.

Second, we also describe a Fuzzy Inference System (FIS) to automatically verify how *good* the composed *SVNs* fit the given customer need. In this way, we can determine to what extent the *SVNs* satisfy the intended customer need. To this aim, for each *SVN*, we analyze the amount of provided, missing and non-required functionalities. Finally, based on their fitness, the *SVNs* are ranked so the customer can select the one better fitting her/his requirements.

This paper is organized as follows. A discussion about basic concepts, problem definition and related work is given in Sect. 2. Sect. 3 describes our composition framework. Finally, we present our conclusions and future work in Sect. 4.

2 Background

2.1 Service Value Networks

We consider a service as an economic activity that offers and requests valuable outcomes to and from its environment [12]. *E.g.* a certain colleague can offer a specific course or certificate in exchange of some tuition fee. *SVNs* are created when different customers and suppliers agree on exchanging valuable outcomes among each other. Based on Hamilton [15], Verna Allee [2], Lovelock and Witz [18], we define a *SVN* as follows:

Definition 1. *A Service Value Network (SVN) is a flexible and dynamic web of homogeneous enterprises and final customers who reciprocally establish relationships with other peers for delivering an added-value service to a final customer.*

The flexibility and the dynamicity within the *SVN* obey to the fact that service suppliers can freely (re)establish relationships with other peers, *i.e.* modifying the structure of the network depending on what a customer requires. By homogeneity we mean that service suppliers have a common business objective, *i.e.* offer a service solution to the final customer. Fig. 1 depicts the basic structure of a *SVN* [5]. Customers exchange valuable outcomes with a pool of suppliers that are arranged into a service bundle bringing about Business to Customer (B2C) relationships. The services within a service bundle can also exchange valuable outcomes with service enablers that support them by means of Business to Business (B2B) relationships [5,21].

The *SVN* in Fig. 1 has been (semi)automatically composed to deal with a specific customer need in the educational service sector [21]. The idea here is to compose *SVNs* for customers that need to improve their job profile by acquiring new skills through some qualifications, certificates or awards offered by educational services. The main motivation is that people looking for a job might have more success by having a better job profile. Furthermore, the skills can be considered as properties, functionalities or according to our concepts *Functional*

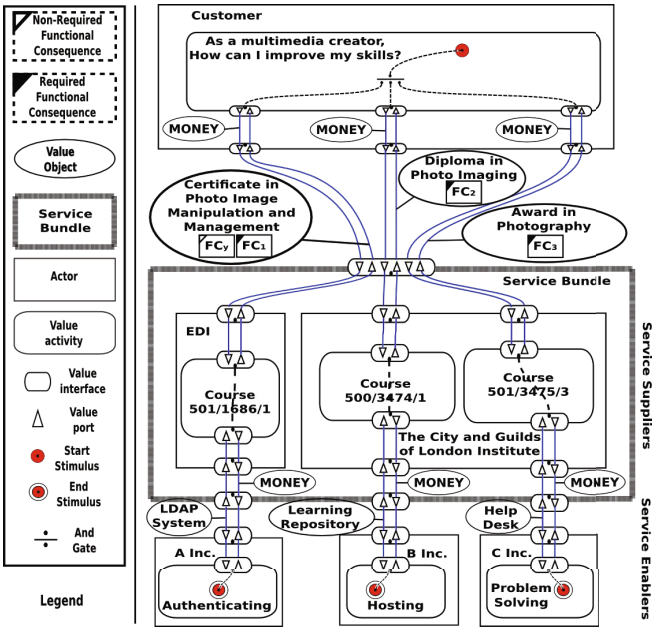

Fig. 1. e^3-*value* model of a *SVN* for a specific customer need requiring FC_1 (Digital Image Manipulation), FC_2 (Photo Image Capture) and FC_3 (Studio Photography). FC_y (Presenting Photo Images) is a non-required skill (functionality) [5,21].

Consequences (FCs) that are offered by an educational service (course) [19,21]. For this case study we harvested the National Database of Accredited Qualifications[1] (NDAQ), which contains details of Recognised Awarding Organisations and Regulated Qualifications in England, Wales and Northern Ireland. The final result was a catalogue containing 58 services offered by four service suppliers.

At the top of Fig. 1 there is a customer that needs to improve her/his job profile by getting new skills related to multimedia creation. In this specific case the customer needs to acquire the following *FCs*: Digital Image Manipulation (FC_1), Photo Image Capture (FC_2) and Studio Photography (FC_3). Furthermore, these *FCs* are packed into the indivisible valuable service outcomes (also known as value objects within the e^3-*value* theory and our framework [12,19,21]): *Certificate in Photo Image Manipulation and Management*, *Diploma in Photo Imaging* and *Award in Photography*. *E.g.* a course offered by a university is composed of a program containing different units that can only be acquired by following the complete course. *i.e.* a valuable service outcome is the smallest unit of economic exchange either its is money or a course.

Since the functional consequences FC_1, FC_2 and FC_3 are not individually offered but contained into indivisible valuable service outcomes that are offered by different suppliers, the only way to obtain these *FCs* is by bundling the

[1] `http://www.accreditedqualifications.org.uk`

services from different suppliers [19]. The advantages of bundling services can be perceived by customers as well as by suppliers. First, from the customer point of view, service bundles are generally cheaper and more suitable for their needs. Second, from the point of view of suppliers, when they work together offering their outcomes, the chances of covering complex customer needs are higher. Third, they can also outsource some of their functionalities to other suppliers/enablers, which not only saves costs of (re)implementation but also allows to focus on and improve their own service outcomes, *i.e.* offering better services [19].

Finally, because service orientation also aims at outsourcing the non-core-business activities from a company, the services inside the bundle may rely on service enablers that perform those activities [18]. *E.g.* As depicted in Fig. 1, since the core business of EDI (Education Development International plc) is providing vocational qualifications, they can outsource services to deal with communication aspects such as a LDAP (Lightweight Directory Access Protocol) system. In the case of The City and Guilds of London Institute, to offer its services it requires the Learning Repository and the Help Desk services. Our framework solves these kind of B2B dependencies by looking for a service that can offer what the service supplier requires [14,21].

2.2 Problem Definition

Contrary to common trends in Service Science that address the service composition problem as a planning problem [10], (semi)automatic composition of *SVNs* can also be addressed from a design perspective. This is mostly due to the fact that business modelling is centred around the notion of value which requires describing *what* the participants exchange with each other rather than *how*, *i.e.* there is no need to deal with the ordering of exchanges. [11].

Nonetheless, to achieve (semi)automatic composition of *SVNs*, three issues must be addressed: *1) Customer-Supplier Interaction (CSI):* To facilitate the co-creation of *SVNs*, customers and suppliers must have the opportunity to completely express their needs and offerings respectively. Moreover, in case *SVNs* do not fit the customer needs, customers must be able of giving feedback to allow the composition of new *SVNs* that can better fulfill their needs. *2) Tailored Composition (TC):* The *SVNs* must be composed based on the functionalities required by a single customer or group of customers which requires alternative $SVNs$ that must meet the customer needs. *3) Automatic Verification (AV):* Once *SVNs* have been composed, we must analyze the individual properties of each *SVN* to determine how well each *SVN* fits the customer need.

We have previously described a framework to (semi)automatically compose *SVNs* [14,19,21]. Since in such a framework we have presented answers for the *customer supplier interaction* and *tailored composition* issues, this paper offers an answer to the third issue, *i.e. automatic verification*. Consider, for instance, the *SVN* depicted in Fig. 1, that offers FC_1, FC_2 and FC_3. Although in this case the *SVN* offers the required *FCs*, sometimes the *SVNs* do not meet the set of required *FCs*. This problem arises because *FCs* are packed within the indivisible

valuable outcomes so the customers cannot get individual *FCs* but a full packet, consequently in some cases more *FCs* are offered than requested (resulting in non-required *FCs*), in other cases less *FCs* are offered than requested (resulting in missing *FCs*). *E.g.*, in Fig. 1 the *Certificate in Photo Image Manipulation and Management* offers FC_1 and FC_y that is not required by the customer.

Furthermore, customers may also have different preferences for each *FC*. Consequently, what is required is: an automatic way to 1) *verify* how well the composed *SVNs* fit the customer needs, *i.e.* how good is the match between the requested functional consequences and the functional consequences offered by the composed *SVNs* and 2) *rank* the *SVNs* so the customer does not get lost with many alternatives. *E.g.*, just in UK, Ofqual has registered around 180 institutions with more than 10,000 services offering different types of qualifications[2]. This generates a very large solution space from which many alternative *SVNs* can be composed.

2.3 Related Work

An in-depth discussion about different service network approaches dealing with *customer supplier interaction* and *tailored composition* can be found at [20], nevertheless, we present a brief overview. There are two main trends about composition within Service Science: 1) Process orientation, and 2) Business orientation. Whereas process-oriented approaches focus on work-flow properties, *i.e.* how the activities must be performed (*e.g.* in which order), business-oriented approaches are centred around the notion of *value*, therefore it is relevant to determine who is offering what of *value* to whom and what expects of value in return, *i.e.* economic reciprocity [11].

Process-oriented approaches are covered by efforts in the field of web services, in which the composition problem is usually modelled as a planning problem and the final solution is given by a sequence of activities to be performed, *i.e.* an execution plan [10]. Some of them make use of meta models or templates [1], others offer more dynamic frameworks [24]. Nonetheless, there are also approaches such as SNN [8] that describes manual composition techniques. Contrary, business-oriented approaches usually rely either on goal or value modelling and the solution is given by a *value proposition* for the final customer [3,6,9].Recently, Becker *et al.* [6] and De Kinderen [9] describe dialogued-based frameworks for service bundling based on predefined bundles. VNA [2] is the only approach that composes a complete SVN however it is also a manual-based approach. Consequently, although process-oriented approaches already offer dynamic frameworks to compose service networks, they lack the value aspects.

Verification in its purest form can be addressed from different perspectives. The scope covers domain-specific calculations or simulations, qualitative simulation and visual simulations [7]. De Kinderen [9] proposes a method called Rank-Order Centroid (ROC) to verify and rank service bundles, nevertheless this

[2] `http://www.ofqual.gov.uk/`

method only focuses on the FCs offered by a service bundle without estimating the impact of missing and non-required FCs which is required when SVN cannot fully cover a customer need.

3 The *e*^{3}*service* Framework

Since the composition task must produce a clear specification about a network of services exchanging different valuable outcomes, methods for designing artifacts can be applied to solve this task [7]. More specifically, the so-called problem-solving methods address design issues from different perspectives such as constraint satisfaction, decomposition of problems, numerical optimization among others [7]. Nonetheless, since *customer supplier interaction* is one of the core issues to achieve SVN composition, we use the propose-critique-modify (PCM) method [7]. As described by Chandrasekaran, a PCM method combines four subtasks (for which different methods can also apply): propose, verify, critique and modify [7].

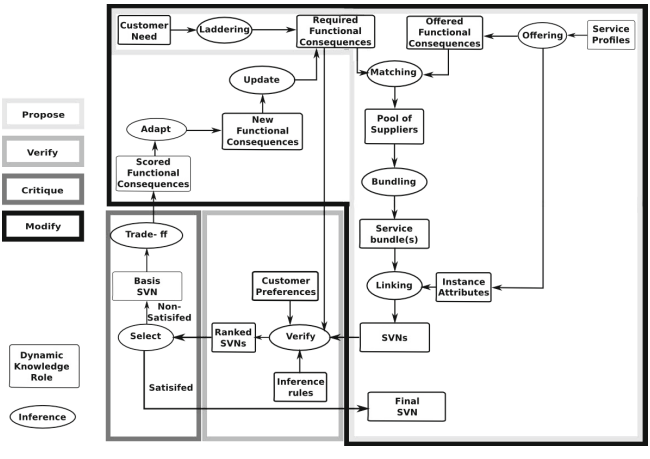

Fig. 2. The *e*^{3}*service* Framework

Figure 2 depicts the main idea behind our PCM method. We use a CommonKADS inference model to describe not only the required knowledge but also the inferences that are needed to produce new knowledge as well as the interactions among them [22]. Inferences carry out reasoning processes whereas dynamic knowledge roles are the run-time inputs and outputs of inferences [22].

In previous work we have implemented part of this framework (mostly the propose task) where supply and demand of services are modelled by means of two ontologies through which suppliers express their offerings and customer express their needs [14,19,21]. Since supply and demand are always evolving, they are described as dynamic knowledge roles. As can be observed in Figure 2, due to the application of inferences, the rest of the knowledge components are also dynamically produced.

3.1 Propose

According to Chandrasekaran, given a design goal, the propose subtask generates a solution [7]. In our case, the goal is to compose a SVN to cover a given customer need. We have extensively explored and developed this subtask in previous work [14,19,21,3,9]. Therefore, we do not present a full description of each inference since this is not the core of the paper. Nevertheless, we briefly explain them as follows:

- **Laddering.** This concept has been widely used in marketing to represent how customers link specific product attributes to high-level values [9]. In our case, by making use of the customer ontology, customer needs as stated by the ontology are refined into so-called functional consequences *FCs*. For instance, a customer need such as "As a multimedia creator, How can I improve my skills?" can be refined into the following *FCs*: Digital Image Manipulation, Photo Image Capture, Studio Photography, Digital Animation and Audio Production among other functionalities that better describe a customer need in terms of specific requirements [9,19,21].
- **Offering.** By making use of the supplier ontology service providers can describe their offerings in terms of *FCs*, *i.e.* what functionalities they can offer to the customers [19,21]. In this way, service offerings can be retrieved to initiate the composition of *SVNs*.
- **Matching.** Because laddering maps customer needs onto *FCs* and service offerings are also described in terms of *FCs*, we can retrieve all the services that completely or partially offer the *FCs* as required by the customer [19,21].
- **Bundling.** Since the output of the matching steps is a pool of service suppliers, at this step we find meaningful combinations of services that can jointly offer an answer to the the final customer [19]. The output is a pool of service bundles that describe B2C relationships in terms of value exchanges.
- **Linking.** At this step we solve the B2B dependencies that service bundles might have [14,21]. For instance, within a bundle, an educational service offering a given course to the customer side might rely on a service such as a digital library that allows students to access reading material. Although the final customer only cares about the B2C with the service bundle, the educational service needs to solve its dependencies with other service enablers to allow the service bundle being sustainable.

3.2 Verify

Also described as *verification* by Chandrasekaran, verify refers to the process of checking whether the design satisfies functional and non-functional specifications [7]. In our framework, however, we only deal with functional requirements. Once the $SVNs$ have been composed, the verification subtask determines not only if the generated $SVNs$ offers the required FCs but also how they fit the customer preferences. For each SVN, three aspects must be analyzed: the *provided*, the *missing* and the *non-required* FCs.

In this way, heuristic reasoning will suggest that a good SVN must have *many* provided, *few* missing and *few* non-required FCs. Moreover, since customers may have different preferences for each FC, we must consider not only whether the $SVNs$ offer the required FCs but also the importance of each FCs. To achieve this task we propose a Fuzzy Inference System (FIS) that can deal with two kind of issues 1) uncertainty: situations in which statements cannot be expressed as false or true but partially true or false, *e.g.* whether a SVN offers the FCs as requested by the customer, 2) computing with words: working with concepts that can be easily understood by human beings, *e.g.* by making use of simple rules we can indicate how the $SVNs$ must be verified [26].

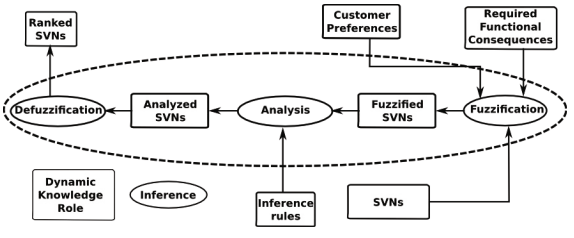

Fig. 3. Fuzzy Inference System (FIS)

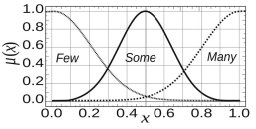

Fig. 4. Fuzzification Functions

As depicted in Fig. 3, our FIS takes as input the required FCs, customer preferences for each FC, the FCs offered by each SVN and a set of inference rules that allow to determine how good the $SVNs$ are. The output is a pool of $SVNs$ that are ranked based on their fitness. The next paragraphs describe each one of the inferences, *i.e.* Fuzzification, Analysis and Defuzzification.

Fuzzification. To determine the amount of provided, missing and non-required FCs for each SVN, we perform three steps: 1) Computing weights, 2) Computing Fractions, and 3) Fuzzification.

1) Computing weights: During the propose phase, customer needs are expressed in terms of *functional consequences (FCs)* that can be denoted as $FC_{Customer}$. The composed $SVNs$ can also be described in terms of FCs highlighting three aspects: 1) FC_P the functional consequences that are *provided* as required by the customer, 2) FC_M the functional consequences that are not offered as required by the customer, *i.e. missing* FCs, and 3) FC_N the functional consequences that are offered by the SVN but are *not required* by the customer. Therefore, in terms of FCs, a SVN can be represented as follows:

$$FC_{SVN} = FC_P \cup FC_M \cup FC_N \tag{1}$$

There are two ways to express preferences for a given FC: w_c and w_a which are defined by:

$$w_c : fc \rightarrow [0, 1], \quad w_a : fc \rightarrow \{0.5\} \tag{2}$$

where w_c is the weight that a customer assigns to a FC and w_a is a predefined weight assigned to a given FC. We have chosen a 0.5 value assuming that non-required FCs have a moderate influence in the fitness of a SVN. Nonetheless, this value can be adapted to the composition context or even gathered through crowd-sourcing. Later on, we can compute the total weights (preferences) for $FC_{Customer}$ and FC_{SVN} in the following way:

$$TW_C = \sum_{fc_i \in FC_{Customer}} w_c(fc_i), \quad TW_{SVN} = W_P + W_M + W_N \tag{3}$$

where TW_C is the total weight for the FCs as requested by the customer and TW_{SVN} is the total weight of all FCs in a SVN. The values for W_P, W_M and W_N are computed as follows:

$$W_P = \sum_{fc_i \in FC_P} w_c(fc_i), \quad W_M = \sum_{fc_i \in FC_M} w_c(fc_i), \quad W_N = \sum_{fc_i \in FC_N} w_a(fc_i) \tag{4}$$

where W_P is the sum of the weights $w_c(fc_i)$ for the functional consequences FCs that are *provided* as required by the customer in a SVN. Similarly, W_M is the sum of the weights for FCs that are *missing* in a SVN, and W_N is the sum of the weights for the FCs that are *non-required* in a SVN.

2) Computing Fractions: For each SVN we compute three fractions: 1) F_P the fraction of the *provided* FCs to the total FCs as requested by the customer. 2) F_M the fraction of the *missing* FCs to the total FCs as requested by the customer. 3) F_N the fraction of the *non-required* FCs to the total FCs in a SVN. These fractions are computed by Eq.5:

$$F_P = \frac{W_P}{TW_C}, \quad F_M = \frac{W_M}{TW_C}, \quad F_N = \frac{W_N}{TW_{SVN}} \tag{5}$$

3) Fuzzification: At this step we determine the amount of *provided* (P), *missing* (M) and *non-required* (N) functional consequences for each SVN in terms of three linguistic labels: *few*, *some* and *many*. To this aim, we compute the membership degree of the fractions F_P, F_M and F_N to the functions in Fig. 4 (also known as Fuzzy Sets) [26]. The membership degree (also known as degree of truth) of a fraction in a fuzzy set allows to determine how *true* is the fact that the analyzed fraction is *few*, *some* or *many*. The degree of truth of a fraction is determined by the following distribution function [26]:

$$Fuzzification(x) = \mu(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\alpha \frac{(x-c)^2}{\sigma^2}} \tag{6}$$

where $x = \{F_P, F_M, F_N\}$, σ^2 is the variance, α is a parameter that determines how width the fuzzy set is and c is the location of the fuzzy set, *i.e.* the highest

point of the fuzzy set. For all the fuzzy sets $\alpha = 2.0$, $\sigma^2 = 0.16$. The values for c are 0.0, 0.5 and 1.0 for the fuzzy sets *few*, *some* and *many* respectively. The values for α, σ^2, c were chosen to cover the fraction values F_P, F_M and F_N within the range $[0, 1]$ [26]. At the end, for each SVN we have different degrees of truth for P, M and N. *E.g.* a SVN with a fraction value $F_P = 0.8$ has the following degrees of truth for P: *few* (0.0), *some* (0.32) and *many* (0.60), which means that P simultaneously belongs to the fuzzy sets *some* and *many* but with different degrees of truth. *i.e.* it is not only true (with a 0.32 value) that the SVN provides *some* FCs but also true (with a 0.60 value) that the SVN provides *many* FCs as required by the customer.

Analysis. Once we have computed the degrees of truth of P, M and M for each SVN, we can then analyze how good they are by making use of a set of inference rules that we designed following common sense criteria and depicted in Fig. 5. *E.g.*, a SVN with $P = some$, $M = few$ and $N = few$ is an *Average* SVN. Since in the fuzzification step P, M and N may have different degrees of truth for each linguistic label (membership function), more than one rule can apply when analyzing a given SVN. *E.g.*, as explained before, P can simultaneously be *some* and *many*, consequently the rules in which P is not only *some* but also *many* have to be applied, the same holds for M and N.

1:	IF	P	is	many	AND	M	is	few	AND	NR	is	few	THEN	Perfect
2:	IF	P	is	many	AND	M	is	few	AND	NR	is	some	THEN	Good
3:	IF	P	is	many	AND	M	is	some	AND	NR	is	few	THEN	Good
4:	IF	P	is	many	AND	M	is	some	AND	NR	is	some	THEN	Good
5:	IF	P	is	some	AND	M	is	few	AND	NR	is	few	THEN	Average
6:	IF	P	is	some	AND	M	is	few	AND	NR	is	some	THEN	Average
7:	IF	P	is	some	AND	M	is	some	AND	NR	is	few	THEN	Poor
8:	IF	P	is	some	AND	M	is	some	AND	NR	is	some	THEN	Poor
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
N:	IF	P	is	few	AND	M	is	many	AND	NR	is	many	THEN	Bad

Fig. 5. Inference rules

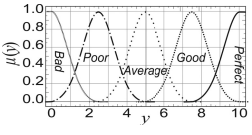

Fig. 6. Defuzzification Functions

To apply the inference rules another observation is important. Since the degrees of truth are expressed in values within the range $[0, 1]$, the traditional boolean AND operation does not work here, *i.e.* after applying the rule (1) as depicted in Fig. 5, IF P is *many* AND M is *few* AND N is *few* the value cannot be 0 or 1 but a value that reflects the degrees of truth of P, M and N. The AND operator in fuzzy logic is usually replaced by the *min* function [26], *i.e.* the output of the rule is the *minimum* degree of truth among P, M and N. *E.g.* for the rule (1), if $P = 0.6$, $M = 0.4$ and $N = 0.6$, the output is *Perfect* with a degree of truth 0.4. At the end of this step, the outputs of all the applied rules are aggregated into a new fuzzy set. Several aggregation methods can be used at this point, in our case we sum up the outcomes for each applied rule and combine them into a fuzzy set given by the membership functions in Fig. 6. The process is analogous to fill a set of silos based on the outcomes of the applied rules (the shape of the silos is given by the functions in Fig. 6). *E.g.* If the rules 2, 4 and 6 (as depicted in Fig. 5) are applied during the analysis and

their value outcomes are respectively *Good* = 0.2, *Good* 0.4, *Average* 0.3, at the end we have a fuzzy set which is the combination of the fuzzy sets *Good* and *Average* but with maximum degrees of truth of 0.6 and 0.3 respectively. The set of membership functions in Fig. 6 is also defined by means of Eq. 6, nonetheless this time the values are $\alpha = 1/8$, $\sigma^2 = 0.16$ and $c = \{0.0, 2.5, 5.0, 7.5, 10.0\}$.

Defuzzification. Based on the fuzzy set generated during the analysis, for each SVN we compute a value that is the final score of the SVN. This defuzzification process can be performed in many different ways. Due to its simplicity, we have used a discrete version of the so-called *center of gravity (COG)* method that easily computes a score and uses the following equation [25]:

$$Defuzzification(A) = D(A) = \frac{\sum_{y_{min}}^{y_{max}} y * A(y)}{\sum_{y_{min}}^{y_{max}} A(y)} \tag{7}$$

where A is the fuzzy set computed during the analysis and Y is the set of elements for which we want to determine a degree of truth with respect to A. Since we want to give scores within the range $[0, 10]$, $Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. To clarify the three steps previously described we present an hypothetical example generated based on the information gathered from the NDAQ database.

Example. Fig. 7 depicts a setting in which three $SVNs$ offer different FCs[3]. SVN_1, that can fully offer the required FCs, is a reduced version of the SVN in Fig. 1 whereas SVN_2 and SVN_3 are networks that can partially cover the customer need. For instance, SVN_3 not only offers FC_2 and FC_3 which are required FCs but also misses FC_1 and offers FC_w that is not required by the customer. Assuming the customer weights in Fig. 7 and applying Eqs. 2, 3, and 4, we obtain the following fractions for SVN_3:

$$F_P = \frac{0.8 + 1.0}{0.6 + 0.8 + 1.0} = \frac{1.8}{2.4} = 0.75, \quad F_M = \frac{0.6}{0.6 + 0.8 + 1.0} = \frac{0.6}{2.4} = 0.25 \tag{8}$$

$$F_N = \frac{0.5}{0.6 + 0.8 + 1.0 + 0.5} = \frac{0.5}{2.9} = 0.17 \tag{9}$$

Afterwards, by making use of the membership functions in Fig. 4, we can compute the values for P, M and N. Since, P is *some* (0.45) and *many* (0.45), M is few (0.45) and *some* (0.45) and N is few (0.70) and *some* (0.25), during the analysis step the rules 1 to 8 are applied as depicted in Fig. 9. As can be observed, the AND operator is replaced by the *min* function and the aggregation stage simply sums up the rules' outcomes.

Once the rules' outcomes have been aggregated, the defuzzification step computes the COG to produce the final score for SVN_3 by means of Eq. 7, with A as depicted in Fig. 9.c, $Y = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. In Eq. 10 we depict how the computation was carried out which can also be visually verified in Fig. 9.c.

[3] For simplicity we only depict the FCs offered by each SVN.

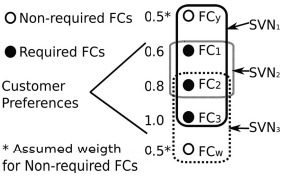

Fig. 7. *SVNs* offering different FCs. Customer requirements are given by FC_1, FC_2 and FC_3. Although SVN_1 offers the requested FCs, it also offers a non-required FC. Moreover, SVN_2 and SVN_3 both only offer two of the requested FCs.

$$D(A) = \frac{1 * 0.17 + 2 * 0.70 + 3 * 0.70 + 4 * 0.45 + 5 * 0.70}{0.17 + 0.70 + 0.70 + 0.45 + 0.70 + 0.45 + 0.80 + 0.80 + 0.45 + 0.45} + \frac{6 * 0.45 + 7 * 0.80 + 8 * 0.80 + 9 * 0.45 + 10 * 0.45}{0.17 + 0.70 + 0.70 + 0.45 + 0.70 + 0.45 + 0.80 + 0.80 + 0.45 + 0.45} = 5.68 \quad (10)$$

Table 1. Important measures in the verification process for each SVN

	SVN_1	SVN_2	SVN_3
P	*Many* (1.0)	*Some* (0.90), *Many* (0.10)	*Some* (0.45), *Many* (0.45)
M	*Few* (1.0)	*Few* (0.10), *Some* (0.90)	*Few* (0.45), *Some* (0.45)
N	*Few* (0.70), *Some* (0.25)	*Few* (1.0)	*Few* (0.70), *Some* (0.25)
Applied rules	1,2	1,3,5,7	1-8
Aggregation	Perfect(0.7),Good(0.25)	Perfect (0.1), Good(0.1), Average(0.1), Poor(0.9)	Perfect(0.45), Good(0.95) Average(0.7), Poor(0.7)
Score	8.69	3.66	5.68

Fig. 9 illustrates how the deffuzification is performed for SVN_1, SVN_2 and SVN_3. Finally, Table 1 summarizes some of the important results that allow computing the final scores for SVN_1 , SVN_2 and SVN_3. As can be observed, for SVN_1 only the rules 1 and 2 are applied whereas for SVN_2 the rules 1,3,5 and 7 are applied.

3.3 Critique

According to Chandrasekaran, if the design task has been unsuccessful, this step identify the source of failure within a design [7]. Furthermore, the main goal at this stage is mostly about finding ways to improve the design [22]. In our case, if a customer is not satisfied by any composed *SVN*, (s)he will identify the sources of failure for a selected *SVN*, otherwise (s)he will select a *SVN* to satisfy her/his need (Fig. 2). More specifically, since *SVNs* might miss *FCs* and/or offer non-required *FCs*, the customers trade off the *FCs* within the selected *SVN*. On the one hand, the customers identify the *FCs* that are not interesting and cause failures to the *SVN*. On the other hand, they can also identify *FCs* that were not required but might be interesting to them. Consequently, in order to achieve this task, two subtasks are required: Select and Trade off.

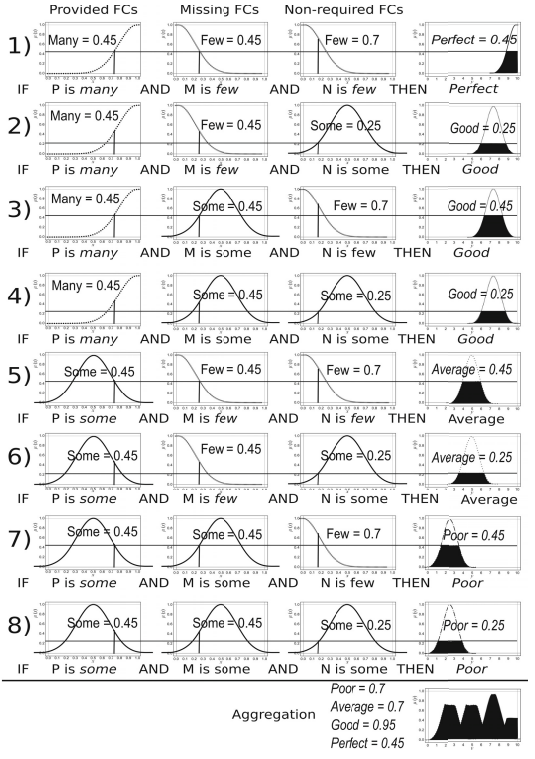

Fig. 8. Applying the first eight inference rules to SVN_3

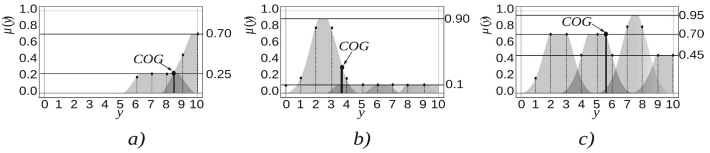

Fig. 9. Defuzzification process for: *a)* SVN_1, *b)* SVN_2, and *c)* SVN_3

- **Select.** Based on the ranking computed by the verification step, the customer can select a *SVN* either to acquire its services or critique its *FCs*. If the customer decides to acquire a composed *SVN*, the goal of the PCM method has been reached. Otherwise, the customer must perform a trade off for the *FCs* offered by the *SVNs*.
- **Trade off.** Once the customer has selected a *SVN* to be critiqued, (s)he must give scores for the *FCs* offered by the *SVN*. The customer is only required to give scores for the *FCs* that were not required, however, (s)he also has the opportunity to change her/his preferences for the previously required *FCs*. With this information, it is possible now to modify the customer requirements and propose new *SVNs* to the customer.

3.4 Modify.

Chandrasekaran defines this step as the process of taking as input information about the failures of a solution design and then changing the design to get closer to the specifications, which involves proposing a new solution design and going again through the verify and critique steps [7]. In our framework, we take as input the customer's feedback to update her/his preferences and propose a new pool of *SVNs* that possibly fit better the customer requirements. Before proposing new *SVNs* two steps must be performed: Adaptation and Update.

- **Adapt.** Based on the scores given by the customer, this subtask can automatically recalculate a new set of *FCs*. The *FCs* with scores greater than zero (0) are part of the new set of required *FCs*, otherwise they are ignored.
- **Update.** Finally, this subtask replaces the old set of required *FCs* by the new set computed in the previous subtask (Adapt). In this way, the composition of new *SVNs* is triggered again, *i.e.* we go back through all the PCM cycle.

3.5 On *e*^{3}*service* Computational Evaluation

The proposed framework has been implemented making use of 1) RDF[4] files to represent customer needs, service suppliers and service enablers, and 2) the Java framework Jena[5] that allows building semantic web applications. To (semi) automatically compose SVNs, we have used 114 services from the NDAQ database. Although the asymptotic complexity of the framework is $O(2^m)$, where m is the number of required FCs, after carrying on experimentation, the performance of the framework remains within reasonable time boundaries (10^4 ms.) for values of $m = [3, 15]$, which is also acceptable since customers rarely request high number of FCs as already observed in previous case studies in the health care, telecomunications and energy industries [16,13,4].

4 Conclusions and Future Work

We have presented a novel framework to compose *SVNs* that is based on the well-known problem-solving method *Propose - Critique - Modify* and provides answers to the *customer supplier interaction (CSI)*, *tailored composition* and *automatic verification* issues. The *e*^{3}*service* framework enhances the co-creation of *SVNs*, by means of *CSI*, which leads to *SVNs* that better fit customer needs. Moreover the composed *SVNs* are not just tailored based on customer requirements but also automatically verified to assess fitness. The main contributions in this work are: 1) a PCM-based framework that contains the needed knowledge to achieve (semi)automatic composition of *SVNs*, and 2) an automatic verification method that determines to what extent the composed *SVNs* fit to the customer requirements.

[4] `www.w3.org/RDF/`
[5] `http://incubator.apache.org/jena`

For the future work we want to include non-functional requirements as well as time and location constraints for the composed *SVNs* so the networks not only offer an accurate answer in terms of functional requirements but also are aware of the quality of the services and the customers' availability and proximity. We strongly consider that the fuzzy-logic-based verification method can easily be extended to face these issues. Finally, we have also to validate more our framework making use of more case studies. Currently, we are exploring composition ideas in the health care sector [9] and global software development [23].

Acknowledgment. This paper has been partially funded by the NWO/Jacquard project VALUE-IT no 630.001.205

References

1. Agarwal, S., Handschuh, S., Staab, S.: Annotation, composition and invocation of semantic web services. Journal of Web Semantics 33, 1–24 (2004)
2. Allee, V.: A value network approach for modeling and measuring intangibles. In: Transparent Enterprise Conference (2002)
3. Baida, Z.: Software-aided service bundling. PhD thesis, Free University Amsterdam (2006)
4. Baida, Z., Gordijn, J., Sæle, H., Morch, A.Z., Akkermans, H.: Energy Services: A Case Study in Real-World Service Configuration. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 36–50. Springer, Heidelberg (2004)
5. Basole, R.C., Rouse, W.B.: Complexity of service value networks: conceptualization and empirical investigation. IBM Syst. J. 47, 53–70 (2008)
6. Becker, J., Beverungen, D., Knackstedt, R., Müller, O.: Model-based decision support for the customer-specific configuration of value bundles. Enterprise Modelling and Information Systems Architectures 4(1), 26–38 (2009)
7. Chandrasekaran, B.: Design problem solving: A task analysis. AI Magazine 11, 59–71 (1990)
8. Danylevych, O., Karastoyanova, D., Leymann, F.: Service networks modelling: An soa & bpm standpoint. J. UCS, 1668–1693 (2010)
9. de Kinderen, S.: Needs-driven service bundling in a multi-supplier setting: The computational e3service approach. PhD thesis, Vrije Universiteit Amsterdam (2010)
10. Dustdar, S., Schreiner, W.: A survey on web services composition. Int. J. Web Grid Serv. 1(1), 1–30 (2005)
11. Gordijn, J., Akkermans, H., van Vliet, H.: Business Modelling Is Not Process Modelling. In: Mayr, H.C., Liddle, S.W., Thalheim, B. (eds.) ER Workshops 2000. LNCS, vol. 1921, pp. 40–51. Springer, Heidelberg (2000)
12. Gordijn, J., Akkermans, J.M.: e3-value: Design and evaluation of e-business models. IEEE Intelligent Systems, 11–17 (2001)
13. Gordijn, J., de Haan, F., de Kinderen, S., Akkermans, H.: Needs-Driven Bundling of Hosted ICT Services. In: van Bommel, P., Hoppenbrouwers, S., Overbeek, S., Proper, E., Barjis, J. (eds.) PoEM 2010. LNBIP, vol. 68, pp. 16–30. Springer, Heidelberg (2010)
14. Gordijn, J., De Leenheer, P., Razo-Zapata, I.S.: Generating service valuewebs by hierarchical configuration: An ipr case. In: Proceedings of HICSS 44 (2011)

15. Hamilton, J.: Service value networks: Value, performance and strategy for the services industry. Journal of Systems Science and Systems Engineering 13(4), 469–489 (2004)
16. de Kinderen, S., Gordijn, J., Droes, R.-M., Meiland, F.: A computational approach towards eliciting needs-driven bundles of healthcare services. In: BLED 2009 Proceedings (2009)
17. Li, M.S.: The Economics of Utility Services in the Future Internet. In: Towards the Future Internet - Emerging Trends from European Research, pp. 31–40. IOS Press (2010)
18. Lovelock, C.H., Wirtz, J.: Services Marketing: People, Technology, Strategy, 7th edn. Pearson Higher Education (2010)
19. Razo-Zapata, I.S., Gordijn, J., De Leenheer, P., Akkermans, H.: Dynamic cluster-based service bundling: A value-oriented framework. In: IEEE 13th Conference on Commerce and Enterprise Computing (2011)
20. Razo-Zapata, I.S., De Leenheer, P., Gordijn, J., Akkermans, H.: Service Network Approaches. In: Handbook of Service Description: USDL and its Methods, pp. 45–74. Springer (2011)
21. Razo-Zapata, I.S., De Leenheer, P., Gordijn, J., Akkermans, H.: Service value networks for competency-driven educational services: A case study. In: 6th International BUSITAL Workshop (2011)
22. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., van de Velde, W., Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press (2000)
23. Tamburri, D.A., Razo-Zapata, I.S., Fernández, H., Tedeschi, C.: Simulating awareness in global software engineering: a comparative analysis of scrum and agile service networks. In: Lewis, G.A., Lago, P., Metzger, A., Tosic, V. (eds.) International Workshop on Principles of Engineering Service-Oriented Systems. IEEE (2012)
24. Traverso, P., Pistore, M.: Automated composition of semantic web services into executable processes, pp. 380–394. Springer (2004)
25. Van Leekwijck, W., Kerre, E.E.: Defuzzification: criteria and classification. Fuzzy Sets Syst 108, 159–178 (1999)
26. Zadeh, L.A.: Fuzzy logic = computing with words. IEEE Transactions on Fuzzy Systems 4(2), 103–111 (1996)