8-1-2010

Networked Mechanism Design Incentive Engineering in Service Value Networks as Exemplified by the Co-Opetition Mechanism

Tobias Conte
Research Center for Information Technology (FZI), conte@fzi.de

Benjamin Blau
Karlsruhe Institute of Technology (KIT), benjamin.blau@kit.edu

Rico Knapper
Research Center for Information Technology (FZI), knapper@fzi.de

Follow this and additional works at: http://aisel.aisnet.org/amcis2010

Recommended Citation
http://aisel.aisnet.org/amcis2010/396

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2010 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Networked Mechanism Design

Incentive Engineering in Service Value Networks as Exemplified by the Co-Opetition Mechanism

Tobias Conte
Research Center for Information Technology (FZI)
Haid-und-Neu-Str. 10-14
Karlsruhe, Germany
conte@fzi.de

Benjamin Blau
Karlsruhe Institute of Technology (KIT)
Englerstr. 14
Karlsruhe, Germany
benjamin.blau@kit.edu

Rico Knapper
Research Center for Information Technology (FZI)
Haid-und-Neu-Str. 10-14
Karlsruhe, Germany
knapper@fzi.de

ABSTRACT

The software industry is currently experiencing a fundamental transition from selling proprietary applications towards the provision of networked services – i.e. modularized and specialized services composed in a plug-and-play fashion in networked service systems. In such service systems, participants’ interests need to be aligned with the network’s global objectives in an incentive engineering approach. In more detail, this paper seeks to tackle the challenge of coordinating self-interested service providers in a co-opetitive environment by designing adequate mechanisms. However, “classic” mechanism design focuses on design goals such as efficiency, incentive compatibility and budget balance whose mere consideration does not always hit the target in service networks. Incorporating the requirements that are imposed by newly arising networked scenarios where a set of agents must co-operate to create value resulting in a complex service, we propose a novel variation of mechanism design: networked mechanism design. Following this approach, we present the co-opetition mechanism as a possible instantiation of networked mechanism design which pursues network-related goals such as network growth, a high degree of interconnectedness, readiness to deliver, and fairness.

Keywords
Mechanism design, service value network, co-opetition, Web services

NETWORKED SERVICES: THE FUTURE OF THE SOFTWARE INDUSTRY

Since the end of the 1990s, the software industry has undergone tremendous changes. Driven by maturing Web services technologies and the wide acceptance of the service-oriented architecture (SOA) paradigm, the software industry’s traditional business models along with business strategies have started to erode – with far-reaching consequences: software vendors turn into service
providers. While traditional software products feature installation on customer site and prepaid perpetual-use licences, so-called software-as-a-service (SaaS) is hosted and maintained by the service provider who offers usage- or subscription-based pricing models. While the success story of SaaS seems to be already sealed, a second wave of innovation has great potentials to shake the software industry’s foundations once again. Exploiting the capabilities of Internet standards and interoperability, joint value creation of service providers comes into play. Open standards and SOA constitute important building blocks for innovative Web service networks, tying together the competencies of specialized contributors while customer value is created via the interplay of complementary service providers.

This new adaptiveness fits the development of software customers demanding more sophisticated as well as more specialized solutions and, at the same time, more flexible service provisioning (Bovet and Martha, 2000). One of the most powerful approaches to handle complexity is modularity, that is composing the whole from smaller subsystems that are designed independently, yet function together as a whole (Baldwin and Clark, 2000). Along those lines, vendors concentrate on their core activity while leveraging knowledge and assets of complementary partners. That way, they are able to stay agile and to flexibly adapt their services to changes in the environment, be it customer-, competition-, or regulation-driven.

However, besides above-mentioned increase in customers’ demands and the resultant agility of service providers, another striking economic factor drives the “servicification” of the software industry: modular services can be combined and (re-)configured into what is known as service mashups. Those have the potential to meet virtually every conceivable customer requirement “out-of-the-shelf”, giving rise to a new level of customization. Such complex services involve the assembly and invocation of several specialized service modules offered by a multitude of specialized partners in order to complete a multi-step business functionality (Papazoglou, 2007). From a technical perspective, dynamic Web services are increasingly used in the context of service mashups, including lightweight approaches such as RESTful architectures and slim messaging formats such as JSON.¹ From an economic viewpoint, value is created through the interplay of various distributed service providers in ecosystem-like environments that jointly contribute to a customized, integrated solution. However, such environments will also include substitutive services and vendors. Thus, service providers find themselves in the fruitful state of co-opetition, breeding both complementary opportunities and competitive threats (Brandenburger and Nalebuff, 1996). While cooperation enables advanced value creation and the access to partners’ assets and knowledge, the competitive component diminishes adverse effects of market power and spurs improvements and innovation.

The above-introduced second innovation wave of the software industry, most notably the power combinatorics in service mashups, is only viable in networked economies. In line with Blau, Krämer, Conte and van Dinther (2009); Krämer, Conte, Blau, van Dinther and Weinhardt (2010), we argue that the potential of the software servicification can be optimally catalyzed in service value networks (SVNs). The operator of such an SVN needs to demonstrate innovative business models and effective incentive schemes in order to pull providers as well as customers onto the platform. Along those lines, rules must be established to implement the set of desired incentives. We claim that “classic” mechanism design, which has proven to be a powerful instrument to solve problems involving self-interested individuals with private information (Mas-Colell, Whinston and Green, 1995), should be extended in its general notion to allow for the inclusion of alternative desired network-based design goals. We pursue such a networked mechanism design approach in our co-opetition mechanism. Capturing both competitive and cooperative elements in SVNs, its twist is to compensate all available service providers that are able to fulfill a specific customer request, not only the ones that are allocated.

The remainder of this article is structured as follows: We continue with a brief definition and formalization of the underlying organizational structure to this article – service value networks. Thereafter, we give a survey on mechanism design and proposed perspectives on it, resulting in the suggestion to abandon some classic objectives in favor of a network-oriented approach. As one instantiation of this newly proposed networked-mechanism design notion, we present our co-opetition mechanism by laying out its objectives as well as its implementation. Thus, the contribution of this article is twofold: First, we motivate a new perspective on mechanism design driven by networked value creation, and second, we present the co-opetition mechanism as a possible instantiation of networked mechanism design.

¹The service mashup platform ProgrammableWeb (http://www.programmableweb.org) reported that by April 2010, 71% of all listed APIs expose REST interfaces, foretelling the trend to an internet of interoperable Web services.
SERVICE VALUE NETWORKS

The smart use of information and communication technology (ICT) as a facilitator to network interaction has lead to new organizational forms, oftentimes referred to as smart business networks. Smartness refers to effectiveness and a comparative advantage through the use of ICT. The latter is also seen as an enabler of network agility, i.e. the network’s ability to “rapidly pick, plug, and play” business processes (van Heck and Vervest, 2007). Service value networks (SVNs) again push these thoughts one step forward facilitating automated coordination and orchestration of complex services by a market platform. Yet it is important to note that the distinction is not drawn by differentiating between products and services. Rather, in line with Vargo and Lusch (2004); Maglio and Spohrer (2008), products can be seen as the vehicles for service delivery as a part of an overall service system.

Consequently, we define service value networks as follows: Service Value Networks are Smart Business Networks that provide business value by performing automated on-demand composition of complex services from a steady but open pool of complementary as well as substitutive standardized service modules through a universally accessible network orchestration platform. (Krämer, Conte, Blau, van Dinther and Weinhardt, 2010).

In order to be able to economically analyze such SVNs, a formalization of their characteristics and components is required. Based on Conte, Blau, Satzger and van Dinther (2009); Blau, Conte and van Dinther (2009), SVNs are formalized by means of a simplified statechart model (Harel and Naamad, 1996).

A service value network is represented by a directed, \( k \)-partite, and acyclic graph. Each partition represents a different functionality requested by the service customer. The set of nodes \( V = \{v_1, \ldots, v_n\} \) represents the set of service offers that are suitable to meet the requested functionality. For the sake of simplicity and without loss of generality, we assume that a service is owned by exactly one service provider such that these concepts can be used interchangeably. Two auxiliary notes, source \((v_s)\) and sink \((v_f)\) act as a makeshift to formalize complex services as an end-to-end connection. Therefore, these nodes are not to be interpreted as services in the network.

According to the different service functionalities that are required in order to reach the overall goal (that is, an instance of the complex service demanded), services are clustered into \( k \) partitions that formalize the specific candidate pools. Substitutive services are mapped to one and the same candidate pool. Consequently, the set of all present candidate pools evolves as \( Y = \{y_1, \ldots, y_k\}, 1 \leq k \leq n \). Exactly one service out of each candidate pool is required to deliver an instance of the complex service requested by the customer. Let \( v_j \in V \) be an arbitrary service in the network. Let \( v_j^m \) denote that vendor \( v_j \) belongs to cluster \( y_m, V^m \subseteq V \) includes the set of nodes allocated to cluster \( y_m \in Y \). Source and sink are, again, also not considered a separate cluster.

An edge \( e_{ij} \) denotes a composition relationship between nodes \( v_i \) and \( v_j \). That is, an edge between two nodes symbolizes the interoperability of the services offered as well as the vendors’ willingness to cooperate. The set of edges shall only include links that are (i) allowed (that is, only links between consecutive clusters) and (ii) available in the SVN.\(^2\) This set of edges shall be denoted by \( E = \{e_{ij}|e_{ij} \text{ is allowed and available in the SVN}\} \). Incoming links of the sink only exhibit an illustrative character and are therefore disregarded in \( E \). Each service \( v_j \) exhibits a service configuration \( A_j \) which aggregates the service’s attributes. In this article, the service configuration is, for simplicity, equalized for all services and can therefore be neglected. Let further \( p_{ij} := p(e_{ij}) \) with \( e_{ij} \in E \) denote the price for service \( v_j \) when being allocated as successor of service \( v_i \).

To summarize, we can formally describe the graph \( G = (V \cup \{v_s, v_f\}, E \cup \{e_{ij}|i \in V^{k-1}\}) \). Technically, let \( G = (V, E) \). An exemplary formalization of \( G \) is given in Figure 1.

Of particular interest in SVNs are instantiable composite service instances as they symbolize a value creating output for the service customer. Only possible realizations of complex services, i.e. complete paths from source to sink, create value. Importantly, a complex service incorporates exactly one service out of each candidate pool. Thus, \( F_i := (W_i, E_i) \) with \( W_i \) denoting a set of services with exactly one service out of each candidate pool \( V^m, m = 1, \ldots, k \), and \( E_i = E(W_i) \) including only those links that connect the services, defines a complex service as one element of the set \( F := (F_1, \ldots, F_h) \) of all \( h \) complex services available.\(^3\)

---

\(^2\)Two services might not be able to interoperete due to technical restrictions or because of strategic considerations.

\(^3\)Since \( v_s \) and \( v_f \) are not considered vendors, we formally exclude them from paths.
MECHANISM DESIGN IN SERVICE VALUE NETWORKS

Why to Apply Mechanism Design in SVNs?

The forerunners of above-introduced service value networks already wait in the wings. Although the market does not exhibit a platform that fully fits our definition, yet several companies have recently launched Web service marketplaces that include a variety of different actors, pointing the way to SVNs by fostering joint value creation of diverse service providers: Salesforce.com offers its on-demand service marketplace AppExchange\(^4\) and its development platform force.com\(^5\), Xignite operates the Splice Mashup Platform\(^6\), and StrikeIron has the IronCloud Web services delivery platform\(^7\) ready, just to name a few. Moreover, companies like Etelos.com with its SaaS Marketplace Platform\(^8\) that offers to other companies frameworks to run Web service marketplaces around their core services push in the market. A very similar concept is successfully marketed by the Chinese platform-as-a-service provider Alisoft.com\(^9\).

Thus, competition is already fierce and will further rise as the SVN concept moves to mainstream. As argued in the introduction, operators of SVNs do not only need to design innovative and elaborate business models but also hold effective incentive schemes ready. The latter is crucial in order to pull participants onto the offered platform and to make sure that they retain there. This is the point where mechanism design comes into play. Mechanism design can be thought of as inverse game theory (Shoham and Leyton-Brown, 2008). It is not about what will happen in a specific interaction of various agents but rather tackles the issue of having a desired outcome in mind and to comprehend which strategic interaction and which setting could lead to a course of action that implements that outcome. This is exactly what an SVN operator must think of when starting a new platform. Besides making a profit in the medium term, which are the short-term goals to pursue to get a business up and running? So, it’s all about incentive engineering – and this is exactly what mechanism design can be ideally applied for. However, mechanism design in its classic form might not always hit the target as we will see in the next section.

Towards Networked Mechanism Design

Before discussing the limitations of classic mechanism design in respect to SVNs, let us quickly go through its fundamentals. The discipline of mechanism design focuses on implementing a preferred system-wide solution to a decentralized optimization problem where self-interest agents act according to their private preferences for different outcomes (Parkes, 2001). The key

\(^4\)http://sites.force.com/appexchange/apex/home/
\(^5\)http://www.salesforce.com/platform/
\(^6\)http://splice.xignite.com/
\(^7\)http://www.strikeiron.com/Company/IronCloud.aspx
\(^8\)http://www3.etelos.com/etelos/platform/
\(^9\)http://www.alisoft.com/
challenge of mechanism design is the actual situation of preferences being private information which induces strategic acting. The agents’ private information cannot be verified by some central authority such as, in our case, a market or platform operator that seeks to achieve certain objectives. Therefore, one has to design a mechanism which establishes some set of incentives, for instance, via side payments, to effectively coordinate participants and eventually enforce the system-wide solution to arrive (Nisan and Ronen, 2001). Such side payments are to compensate the agents for potential individual disadvantages that are in place if the desired result occurs. In other words, mechanism design is to implement institutional rules that determine decisions as a function of the information that is known by the individuals in the economy, thereby ensuring desired events to occur even if players act opportunistically and may try to strategically manipulate in order to maximize their individual utility (Myerson, 1988). Above-mentioned function is called the social choice function. Generalizing the notation introduced in the previous chapter, let us denote the set of all agents considered by \( I = \{1, \ldots, n\} \), each of them equipped with certain preferences: her type \( \theta_i \in \Theta_i \), \( \theta = (\theta_1, \ldots, \theta_n) \in \Theta \). The social choice function \( g : \Theta_1 \times \cdots \times \Theta_n \to H \) chooses the system-wide preferred goal (or outcome) \( g(\theta) \in H \) with \( H \) being the set of all possible outcomes.

Each agent \( i \in I \) has available a set of possible actions or strategies \( \Omega_i \) summing up to \( \Omega = \Omega_1 \times \cdots \times \Omega_n \) over all agents. Assuming quasi-linear preferences and risk neutrality of the agents, a mechanism \( \mathcal{M} \) is, simply spoken, a triple \( (\Omega, o(\cdot), t(\cdot)) \) where the outcome of the mechanism is set up by an allocation function \( o : \Omega_1 \times \cdots \times \Omega_n \to H \) that maps agents’ strategies to an outcome \( \eta \in H \) and a transfer function \( t : \Omega_1 \times \cdots \times \Omega_n \to \mathbb{R}^n \) with \( t(\cdot) = (t_1(\cdot), \ldots, t_n(\cdot)) \) that determines the payment made or received by each agent \( i \in I \) (Mas-Colell, Whinston and Green, 1995; Parkes, 2001).

**Classic Mechanism Design.** Classic mechanism design originates from the seminal works provided by Vickrey (1961), Clarke (1971), and Groves (1973), being namesake to the prominent class of Vickrey-Clarke-Groves (VCG) mechanisms. VCG mechanisms are second-price-sealed-bid mechanisms that focus on enforcing incentive compatibility (IC). In incentive compatible mechanisms all agents are incentivized to reveal their true type as an equilibrium strategy. Besides truth-telling, the focus of mechanism design has been put upon three other desiderata that are closely related to IC. One of them is allocative efficiency (AE), denoting that a mechanism’s social choice function always determines an outcome \( \eta^* = g(\theta) \) such that there is no other outcome \( \eta' \in H \) which yields a higher valuation for all agents. Put differently, \( g(\theta) \) is allocatively efficient if \( \eta^* \) maximizes the total value over all agents (i.e. the system’s welfare). Further, individual rationality (IR)\(^{11}\) makes sure that agents do not suffer any loss by participating in \( \mathcal{M} \). Finally, budget balance (BB) denotes that distributes and collects the same amount of money to and from the agents\(^{12}\), that is, no outside payments are required to realize \( \mathcal{M} \) (Shoham and Leyton-Brown, 2008).

As academics have proven, impossibilities exist among these desiderata. The most seminal one was shown by Myerson and Satterthwaite (1983), stating that if assuming quasi-linear preferences, there is no double-sided mechanism to achieve AE, weak BB, ex interim IR, and Bayesian Nash IC at the same time. Thus, above-introduced properties must be traded-off in some way. For example, VCG mechanisms can lead to serious subsidiaries made by the mechanism (Parkes, Kalagnanam and Eso, 2001). Vice versa, individually rational and budget balanced mechanisms can be highly inefficient (McAfee, 1992; Barberà and Jackson, 1995).

**Second-Best and Algorithmic Mechanism Design.** Above-stated inefficiency is the point where the so-called approach to design second-best mechanisms according to Parkes, Kalagnanam and Eso (2001) becomes an issue. To grasp the idea of such second-best-mechanisms note that it is possible to construct inefficient, but incentive compatible mechanisms\(^{13}\). However, AE inherently requires IC. Only if the latter is ensured, a globally efficient result can be achieved and verified. However, as Parkes, Kalagnanam and Eso (2001) argue, non-incentive-compatible mechanism design as a variation of second-best mechanism design can actually be useful, although truthful bidding is not an equilibrium strategy for agents and thereby, AE is also sacrificed.

Certainly, mechanism design in the narrow meaning does not allow for – or, formulated more precisely – does not require such a variation: in theory, non-incentive-compatible mechanisms can be subsumed under the class of incentive compatible mechanisms.

\(^{10}\)There can be different underlying equilibrium concepts, e.g. an equilibrium in dominant strategies or a Bayesian Nash equilibrium. The interested reader is referred to Mas-Colell, Whinston and Green (1995).

\(^{11}\)IR can be based either on the agents’ expected utility (ex interim IR) or their ex post utility (ex post IR).

\(^{12}\)If net transfers can be made from agents to the mechanism (but not the other way round), one speaks of weak BB.

\(^{13}\)That is, IC does not influence the impossibility statement made by Myerson and Satterthwaite (1983).
ones. According to the revelation principle, any social choice function realized (in equilibrium) with a non-incentive-compatible mechanism can be transferred in an equivalent incentive compatible mechanism where every agent directly “plays” its true type (Myerson, 1979). However, this is the crux, computational assumptions of the revelation principle are unrealistic. First, it assumes that agents in the non-incentive-compatible mechanism are capable of computing any of their equilibrium strategies. This is an implicit worst-case assumption of mechanism design on agents’ strategic abilities, saying that agents can always compute and exploit all opportunities for manipulation present in the considered mechanism. Second, for the submission of agents’ strategies and the computation of the outcome by the mechanism operator, the revelation principle postulates unlimited computational resources (Parkes, 2001; Parkes, Kalagnanam and Eso, 2001).

Yet, in selected applications, from an economic standpoint, it might not be the ultimate goal to achieve a truthful revelation of the agents’ types “at any costs”. Sacrificing IC in favor of other properties can be reasonable in order to obtain a “good (enough)” result. Even if incentive-compatibility is not given, the agents’ behavior can still be predictable in a sense that the social choice is met in a broader sense – however, then also dismissing AE. If IC is to be enforced, inefficient solutions are deliberately accepted – and such inefficiencies can be enormous as IC and BB/IR mechanism implementations give proof of (Parkes, Kalagnanam and Eso, 2001). Yet, it is possible to achieve less inefficient allocations without insisting on truthful information revelation.

Closely related to the revelation principle’s assumption of unlimited computational resources (and parallel to the second-best-mechanism approach), computational or algorithmic mechanism design (AMD) came up in the late 1990s, aiming at the handling of the complexity exhibited by many Internet-based applications of mechanism design such as routing problems (Nisan and Ronen, 2001). First and foremost, determining the allocation rule $o(\cdot)$ and the transfer function $t(\cdot)$ of $\mathcal{N}$ can cause severe complexity and is usually referred to as the weightiest and “most costly” criterion. Therefore, tractability comes into the mix. A mechanism is said to be tractable if its allocation rule and its transfer function can be computed in polynomial time. However, among others, the famous VCG mechanism is computationally intractable. Therefore, the tension between computational tractability and desirable game-theoretic properties can lead to a similar result than above-motivated second-best-mechanisms: by approximating $o(\cdot)$ and/or $t(\cdot)$, the mechanisms’ tractability might be restored at the price of breaking IC and therefore AE (Parkes, Kalagnanam and Eso, 2001). To summarize, AMD sacrifices economically efficient outcomes in order to guarantee computational efficiency.

**Networked Mechanism Design.** Building upon the approaches presented in the previous section, we move one step ahead, postulating what we call the networked mechanism design perspective. Unlike second-best mechanism design, we do not only sacrifice classically applied desiderata in order to approximate other classic properties. Networked mechanism design is rather geared to the very idea of AMD: to sacrifice certain desiderata in order to achieve properties that arise from the underlying application domain. However, we do not keep focus on tractability, but incorporate the requirements that are imposed by newly arising networked scenarios where a set of agents must co-operate to create value. At the same time, however, agents compete not only for an inclusion in the complex service offered, but also for their shares in the revenue that arises from the joint value creation. This area of conflict is known as co-opetition, which is inherent to problem sets in networked scenarios such as incentive engineering in SVN. Designing a business in SVN, alternative properties related to a healthy network development appear on the scene as we will show in the remainder of this article.

Two concrete consequences applying networked mechanism design must be paid attention to. First, the nature of network-related properties are potentially likely to differ from classic desiderata. While the latter can be formulated on a very high level of granularity, properties of networked mechanism design can also take over the characteristic of a more globally formulated target setting rather than a desideratum. Aberrant from classic mechanism design, target settings might be formulated in relative terms. Such relative verbalized objectives requires a comparison to suitable benchmarks. Second, akin to AMD and second-best mechanism design, classic design desiderata are likely to fall victim to the alternative goal pursued.

In the next section, we provide a mechanism sketch formulated according to the networked mechanism design approach – the *co-opetition mechanism*. Due to space restrictions, the mechanism is stripped to its most important features.

---

14 Focusing on network-related requirements, computational issues are put aside in this article.
THE CO-OPETITION MECHANISM

As indicated in the introduction, the operator of an SVN needs to attract both service providers and service customers with both sides valuing participants on the other market side. With either market sides being initially vacant, we face a typical chicken-egg-problem. We follow the approach to start with setting incentives for the service providers side which is then assumed to generate positive feedback upon the service customers and so forth.

Social Choice

In order to capture an SVN’s co-opetitive nature, networked mechanism design goals predominate when defining a system-wide preferred goal. Taking the platform operator’s view in the launching phase of an SVN, fueling initial business must be the main goal. The key objective is to define incentives that activate network effects and open out into positive feedback loops giving rise to self-reinforcing success.\(^{15}\)

Two central and crucial objectives shape the mechanism proposed that stand above profit making in the short run. First, it is the mechanism’s ability to set effective incentives for participants to join, thereby fostering network growth. Second, such growth shall be sustainable, therefore service providers should be compensated for constantly and continuously keeping ready their services in the network. We refer to this goal as readiness. If we want to compensate all service providers in the SVN that generate added value, the distribution logic must be fair. On the one hand, to retain competition, it is obvious that service providers that make greater contributions to the aggregate system’s overall value need to receive a greater share of. On the other hand, a distribution logic needs to be conceived as generally “evenhanded” by participating providers in order to gain acceptance. Finally, the co-opetition mechanism shall account for a high degree of interconnectedness among service providers. Promoting alternative paths through the network leads to a more balanced network without single providers having monopolistic positions. In such balanced networks, the platform operator is no longer dependent on powerful service providers which could impose pressure by bulling the market or by threatening the network with termination of membership. Further, as known from network economics, a highly interconnected network is a prerequisite for complementarity (Economides, 1996).

In addition of the design goals that arise due to the network-centric interaction of the participants, we require the co-opetition mechanism to fulfill BB and (ex interim) IR to make it sustainable. Individual rationality is vital since agents are not willing to voluntarily participate if they expect to incur losses. On the other hand, a mechanism cannot be continuously subsidized by its operator in the medium-term (Parkes, 2001).

Mechanism Implementation

Recall that an SVN is formed upon specific customer request – any complex service in the SVN is thus potentially able to create value. Even if not allocated, each and every service enriches the platforms variety and makes a contribution to the overall SVN. Therefore, it is may lead the desired results to compensate service providers that have designed their offerings according to the SVN’s requirements also for their readiness to deliver.

Allocation Function. We formulate the allocation function as a maximization problem that maps service providers’ price bids \(p_{ij}\) and the service customer’s request willingness to pay \(\alpha\) – both elements denoting their reported types\(^{16}\) (i.e. actions) – to a feasible and optimal path \(F^*_l \in F\). The utility \(U_{F_l}\) a complex service \(F_l\) creates in the SVN is defined by the customer’s willingness to pay net the sum of the price bids of services included in \(F_l\):

\[
U_{F_l} = \alpha - \sum_{e_{ij} \subset F_l} p_{ij}
\]

As we seek to align the distributed surplus to the value service providers create for the system, it suggests itself that we choose \(\max_{F_l \in F} U_{F_l} = U_{F^*_l}\) as the monetary resources that available for distribution.

\(^{15}\)Profit maximization of the platform operator is not a primary aim at this stage of the business and is therefore not pursued in this work. It is rather important to support and activate a healthy and preferably speedy network development.

\(^{16}\)Recall that the co-opetition mechanism is simplified in this article. In its extended form, it also takes account of service configurations and the customers’ preferences for service attributes.
In order to determine the optimal path, we choose the complex service that maximizes the value for the system as shown in the following equation:

\[
o := \text{argmax}_{F_j \in F} \left( \alpha - \sum_{v_j, e_{ij} \subseteq F_j} p_{ij} \right) \quad \text{s.t. } U_{F_j} \geq 0 \forall F_j \in F
\]

(2)

The constraint \( U_{F_j} \geq 0 \forall F_j \in F \) is required to guarantee individual rationality of the customer and budget balance at the same time.

**Transfer Function.** The novelty of the payment function \( t(\cdot) \) is to both apply a purely allocation-based component \( (t^1) \) and a component that takes account of the overall marginal contribution of each service to the specific customer-driven SVN – the power ratio \( (t^2) \). In order to specify \( t^2 \), we introduce the concept of internal cooperations. Given the set of all bids that comprise \( G \), the set of internal cooperations is defined by the power set of \( V \), each element \( V_m \) attached with its actual links \( E(V_m) \). In more detail, let \( S := (S_1, \ldots, S_{|p(V)|}) \) with \( S_m := (V_m, E_m) \). Incorporating Equation (3) and the concept of including each sub-network \( S_m \in S \) of \( G \), Equation (4) yields the power ratio of service \( v_j \):

\[
\phi_j(G, \chi) = \sum_{S_m \in S \mid v_j \in S_m} \chi(S_m) - \chi(S_m \setminus \{v_j, E(v_j)\})
\]

(4)

with

\[
\gamma_{S_m} = \frac{(|V_m| - 1)! \cdot (|V| - |V_m|)!}{|V|!}
\]

(5)

The set of all reasonable linkages of a service \( v_j \) within a cooperation \( S_m \in S \) is denoted \( E(v_j) \). As soon as a service \( v_j \) is included in a cooperation \( S_m, E(v_j) \) is also added.

Consequently, the power ratio-based transfer function (PRTF) in respect to service \( v_j \) consists of a directly allocation-dependent component \( t^1_j \) and a component \( t^2_j = \phi_j \):

\[
t_j = t^1_j + t^2_j := \begin{cases} p_{ij} + \phi_j, & \text{if } v_j, e_{ij} \subseteq E^* \\ \phi_j, & \text{otherwise} \end{cases}
\]

(6)

\(^{17}\)Recall that \( E(V_m) \) stands for the set of all edges that are associated to the set of services \( V \) that are potential service candidates in the SVN.

\(^{18}\)For all internal cooperations \( S_m \in S \) a service \( v_j \) can (theoretically) be part of, the rightmost term of Equation (5) takes a positive value whenever \( v_j \) is pivotal to \( S_m \). It measures the service’s marginal contribution to the considered internal cooperation. This value is weighted by the probability of the underlying cooperation to form \( \gamma_{S_m} \) consulting the logic introduced by SShapley (1953). For more details on the Shapley value, the interested reader is referred to Mas-Colell, Whinston and Green (1995).
CONCLUSION & RELATED WORK

This article is building upon “second-best mechanism design” (Parkes, Kalagnanam and Eso, 2001) and algorithmic mechanism design (Nisan and Ronen, 2001) as variations to classic mechanism design. These approaches led the way to a notion that a waiver of traditionally pursued objectives such as incentive compatibility and allocative efficiency can be useful in certain application domains. In this vein, by presenting the co-opetition mechanism, we showed an application of networked mechanism design in which network-related objectives overbalance the desiderata from classic mechanism design.

Due to space restrictions, we cannot show in detail that the co-opetition mechanism fulfills the social choice as introduced above. While readiness is inherently fulfilled by applying Shapley-style calculus (cp. Equation 4), readers are pointed to earlier publications showing that the other objectives are met as well. In Conte, Blau and Xu (2010) we showed the co-opetition’s ability to incentivize network growth. The mechanism’s potential to account for a high degree of interconnectedness was evaluated in Conte, Blau, Satzger and van Dinther (2009). In more detail, the PRTF accounts for significantly stronger incentives for participants to join an SVN and leads to significantly increased link building efforts compared to other transfer functions that merely reward allocated service providers. Finally, a proof sketching the fairness requirement can be found in Conte, Blau, Satzger, van Dinther and Weinhardt (2009). One can show that the fairness properties that are originally featured by the Shapley value (Shapley, 1953) are also valid for the PRTF.

By virtue of the co-opetition mechanism’s roots in both cooperative and non-cooperative game theory as well as its borrowings from network theory, related work and approaches stem from diverse disciplines that can be roughly classified into three categories. First, mechanism design approaches in relevant domains have been presented by, for instance, Blau, van Dinther, Conte, Xu and Weinhardt (2009) and Mohabey, Narahari, Mallick, Suresh and Subrahmanya (2007). They include complex services as object of network-based trading, however, do not explicitly consider requirements that arise from the underlying network structure. Second, multiattribute procurement auctions match our approach in terms of the auction type. Again, work proposed by Parkes and Kalagnanam (2005) or Ronen and Lehmann (2005) either deal with classic mechanism design properties or AMD, neglecting networks and their structure. Finally, allocation rules from network design are related to the co-opetition mechanism in terms of value distribution: approaches presented by Myerson (1977); Jackson and Wolinsky (1996) and Jackson (2005) grasp network-related properties, yet neither fit the application scenario of SVNs nor the discipline of mechanism design. The co-opetition mechanism brings together above-listed approaches in an integrated effort to design a mechanism that is in line with network-related properties in a co-opetitive environment – the service value networks.

ACKNOWLEDGEMENTS

The research was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference “01MQ07012”.

REFERENCES


