Please quote as: Li, M. M.; Peters, C. & Leimeister, J. M. (2018): A Hypergraph-based Modeling Approach for Service Systems. In: INFORMS International Conference on Service Science. Phoenix, Arizona, USA.

A Hypergraph-based Modeling Approach for Service Systems

Mahei Manhai Li[*1], Christoph Peters[1,2], and Jan Marco Leimeister[1,2]

[1]Information Systems
Research Centre for IS Design (ITeG)
University of Kassel,
34121 Kassel, Germany

[2]Institute for Information
Management
University of St.Gallen
CH-9000 St.Gallen, Switzerland

ABSTRACT

Currently, research on service science has emerged as its own discipline, where service systems are its basic unit of analysis. However, without a clearly defined modeling approach for service systems, analyzing a service system is challenging. We therefore propose a conceptual hypergraph-based modeling approach, which can be used to model services for both traditional goods-dominant businesses, as well as service-businesses. We define key elements of a service system, while drawing upon hypergraph theory and present three modeling properties which are required to model a service systems graph (SSG). The focus of SSGs is to describe the relationships between the various resources, actors and activities, thus configuring a service system. It provides the foundation for computer graphic simulations and database applications of service business structure for future research.

Key words: Service Systems, Service System Graphs, Modeling, Service Modeling, Service System Modeling, Service Systems Engineering, hypergraph

1. INTRODUCTION

Throughout the emergence of Service Science, service systems have always been a key concept of the discipline (Spohrer et al. 2007). Since the seminal paper on service dominant logic (Vargo and Lusch 2008) both service-centric businesses and traditionally goods-dominant businesses have begun to apply a service perspective on their organization in order to remain competitive and innovate (Lusch and Nambisan 2015). Especially research on the transition to services has gained traction by terms, such as servitization (Howard Lightfoot et al. 2013).

Service as a way of thinking has gradually evolved and is used by both manufacturing and service-businesses, since production can be also be seen as internal services for providing an end-customer a value proposition (Hill 1977). Thinking in service systems can help identify service innovation potentials (Beverungen et al. 2018; Breidbach and Maglio 2015). However, we model service systems using a multitude of modeling approaches focusing on actors and a processual perspective (Bitner et al. 2007; Lim and Kim 2014; Patricio et al. 2011; van Eck et al. 2009) or more technical perspectives. Yet, it can be challenging for practitioners to utilize the concept of service systems from a business perspective (Alter 2017). Since service scientists "study, manage, and engineer service systems, solving problems and exploiting opportunities to create service innovations" (Maglio et al. 2006), our research goal is to provide a tool to model and analyze service systems. Our research question is therefore as follows: How can we model basic service systems both correctly and graphically?

2. A SERVICE SYSTEMS PERSPECTIVE

The service system's inherent focus lies in finding the right configurations of resources for actors in order to create value in the right context (value-in-context, formerly referred to as value-in-use) through the use of services to customers (Böhmann et al. 2014; Leimeister 2012; Vargo and Lusch 2008). Vargo and Lusch (2008) have addressed the configuration in applying their concept of service dominant logic and called it resourcing. A service system is

[*] Corresponding author: +49-561-804 6046; fax: +49-561-804 6067; E-Mail: mahei.li@uni-kassel.de

guided by a value proposition, which in turn has a corresponding configuration of actors and resources (Böhmann et al. 2014; Vargo and Lusch 2008).

We define a service system as a value cocreation configuration of resources (Maglio and Spohrer 2008). This perspective is rooted in service dominant logic (Maglio and Spohrer 2013). Resources include both operand and operant resources (Maglio et al. 2010). Different configurations of resources are connected by respective value propositions, sometimes also seen as service exchanges (Vargo and Akaka 2012).

Recent research revisits the importance of value propositions and engagement of service systems (Chandler & Lusch 2015), in which organizations seek to find the right constellation of actors ("who") within a service system that enables actors to find the correct resources ("who" and "with whom") for a specific context ("when") in order to co-create value (Chandler & Lusch 2015, p.1), whereas the creation of value ("value-in-use") happens through activities between actors, also referred as interactions (Vargo and Lusch 2008). This coincides with an input-output perspective, in which the realization of value happens through a transformation process of resources by actors (Hill 1977).

The actors are essential to realize the initially proposed value. They act upon the resource configurations to achieve the value proposition. Since a service system includes different types of resources and actors, who create value to a customer, we define the term „service objects "that pairs corresponding resources and actors, for a value proposition. Realizing the value proposition for the customer is imperative. From a service-provider perspective, it is decisive to know the constellation of resources that actors require. An actor can be individuals, teams, organizations or even software systems, if they mobilize the required resources. The service system therefore needs to be orchestrated to bring all resources and actors together. Hence, our service system graph is focused on service orchestrators as key stakeholder.

In conclusion, the constituent elements of a service systems are: resources, actors, service objects and activities. These elements should be configured to realize value. These elements serve as form of lightweight ontology for our service system modelling approach. Key contribution of this paper is the definition of their relationships using hypergraph theory.

3. DEVELOPING SERVICE SYSTEMS GRAPHS (SSG)

First, we define the key concepts of our service system using hypergraph theory, which has its origins in graph theory and generalizes upon the concept of graphs (Berge 1989). A hypergraph G=(V, E) exists as a pair of edges E and set of vertices V, where the edges $e \in E$ does not only connect two, but any number of vertices $v \in V$, thus calling E a set of hyperedges. A hyperedge $e \in E$ is therefore a subset of all vertices V, which are connected by it, $e \subseteq V$. Additionally, E is a subset of $P(V) \backslash \emptyset$, where P(V) is the power set of V.

Since service systems require resources as input factors, we define a set R with $r \in R$ as all required forms of resources. Service systems also require actors (Breidbach and Maglio 2016). We define actors as a set A with $a \in A$ representing an actor. Since we define A as the set of required actors, it would be better to consider A as a team or organizational unit that is required for providing the service. An actor a can thus be an individual, a group, an organizational unit or even software systems.

A service system for a specific value proposition requires both actors and related resources. We called a pair of actor and required resources, "service objects" and define all service objects as a set O with $o \in O$ being a single service object. Formalized, a service object is a tuple of the required resources and the required actors specific to a value proposition. Hence, service objects are the subject matters of service systems, which are defined in a specific context as input sets of respective outputs. Let $O \neq \emptyset$ be the set of required service objects of any service-driven organization, with $o \in O$ defined as a service object. Thus, a service object is a tuple consisting of resources and actors. Formalized, service objects are defined as follows:

Definition 1

A finite non-empty set O with tuple of (R, A) is called service object where

i. R is a finite set of resources with $R=\{r_1, r_2 \dots r_n\}$;

ii. A is a family of subset actors of R with $A=(a_i)$ in which $a_i \subset R$ and $R=\bigcup_{i=1}^{n} a_i$ for $i \in \{1,2...n\}$.

Definition 1 shows that service object O is essentially a hypergraph (Berge 1989). Therefore, the service object O, the tuple (R, A) is a hypergraph of service objects, which inherently represent all possible value propositions of said service system. In other words, the potential of a service system can be unlocked by reconfiguring its resources and

assigning it a suitable actor.

Hypergraph theory has extensively focused on its sets of vertices (Bretto 2013) , whereas we put equal importance to its hyperedges. Due to the roles of actors in service science, we inscribe the semantic meaning of actors into hyperedges. A service object includes both actors and resources, both paramount for the realization of the service.

The service object is the static part of a service system. It constitutes the necessary input resources R, which actors A require, before an actor can provide value to a service consumer. In other words, it represents the potential value an actor can provide to a potential consumer.

The vertices (sometimes known as nodes) of a hypergraph G_i represent resources and hyperedges of G_i represent actors, whereas we define required actors and resources as service objects. Following hypergraph theory, hyperedges can intersect with each other, illustrating shared resources.

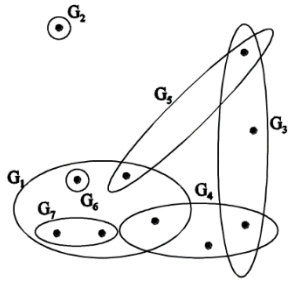


Figure 1: Hypergraph G

The service object can be the end result, as well as intermediate results of any service, each representing value propositions in terms of service exchanges. If the above mentioned elementary object is put together with other service objects, another service system can be configured. This is an analogous characteristic to "traditional" manufacturing cases (e.g.: Hill, 1977), in which outputs are used as inputs for other processes, thus creating a path. We will revisit the path characteristic shortly, when introducing service activities.

An element graph is a graph of order = 1, that is, $|G_i|=1$ for $i \in \{1,2...n\}$ (Berge 1989). It represents a service object $o_0 \in O$ with tuple (a_0, R_0) where $|R_0|=1$. It is apparent that the elementary graph itself has edges. We changed the representation from a solid dot by adding a circle around it to indicate that it also has an hyperedge and hence constitutes a service object, as depicted in Figure1. We argue that single resources can always be considered as element graphs. However, we recommend only drawing the hyperedge if it is either an explicit output of a service object or if the element graph itself is a single input.

Although we have mapped service objects, which consist of actors and resources, we have yet to map a sequence of activities into our modeling approach. Up until now, a hypergraph can be used to model non-directional set of elements that contains the information of relationships among elements with the help of hyperedges. To map the relationship of elements of different hypergraphs, directional hypergraphs can be used to map the relationship of elements towards other elements of different graphs (Gallo and Scutellà 1998). However, it is not possible to map entire hypergraphs towards other hypergraphs or toward elements of other hypergraphs. We need to expand upon the existing definitions of hypergraphs. We do so by introducing an approach to map a service object to other service objects with ψ. In the following section we will define how to map hypergraphs to other hypergraphs. This enables us to model service systems with hypergraphs.

Definition 2

O is a finite non-empty set of service object and O is a hypergraph of service objects. A mapping $\psi(\psi^+,\psi^-)$

with $\psi: 0: O \times O \rightarrow$ Boolean where $O \times O \subset 2^O$

is called a service activity of service objects.

Service activities for service objects are represented by the binary mapping between different service objects. One service object is seen as input, whereas the other is seen as output, while the value realization is enabled by an activity that makes the transition from one service object to another possible. The mapping ψ is a tuple of (ψ+,ψ-), which is a directed or counter-directed mapping of hypergraphs. In this paper, ψ and ψ + are used synonymously for directed mapping (Figure 2), accompanied by the drawing of an arrow line. This is not to be confused with directed hypergraphs,

which only allows relationships between elements of different hyperedges (Gallo and Scutellà 1998).

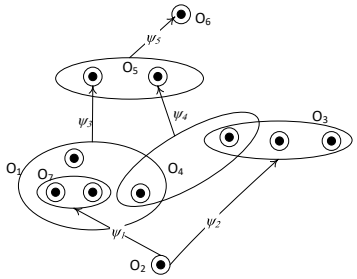


Figure 2: Directed Mapping of Hypergraphs

Definition 3:

Let R be a finite nonempty set of resources, A a finite nonempty set of actors and O a set defined as tuple (R, A) be a hypergraph of a service object and Ψ be a set of functions as service activities. Then the tuple (R, A, Ψ) is called the SSG or service system graph,

where $\Psi:\Psi(O) \rightarrow \Psi^{+}(O)$ with $\exists\, o \in O \mid \Psi^{-}(o) \cap \Psi^{+}(o) = \emptyset$.

Function Ψ(O) defines which service objects are required as input factors and function Ψ+(O) defines the output service objects.

A service system graph is a directed graph, which models the value creation and value propositions of a chain of services. The service system is a family of subset service objects. Thus, strictly speaking, a single service object itself is also a service system.

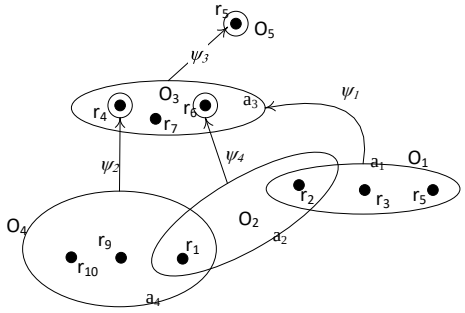


Figure 3: Example Service System Graph with $R_i \Leftrightarrow O_i$

To illustrate the relationships of a service system, we present the detailed example of a SSG (R, A, Ψ): Figure3 shows the SSG with a set of resource R={ r_1,r_2....r_{10}}, a family of the subset A=(a_1, a_2, a_3, a_4, a_5); $a_1=R_1=\{r_2, r_3, r_5\}$; $a_2=R_2=\{r_1, r_2\}$; $a_3=R_3=\{r_4, r_6, r_7\}$; $a_4=R_4=\{r_1, r_9, r_{10}\}$; $a_5=R_5=\{r_5\}$; and the function $\Psi=(\psi_{1,}\psi_{2,}\psi_{3,}\psi_4)$ where $\psi_1=((a1,\{r_2, r_3, r_5\}), (a_5\{r_4, r_6, r_7\}))$; $\psi_2=((a_4,\{r_1, r_9, r_{10}\}), (a_0, \{r_4\}))$; $\psi_3=((a_5\{r_4, r_6, r_7\}), (a_0, \{r_8\}))$; $\psi_4=((a_3,\{r_4, r_6, r_7\}), (a_0\{r_5\}))$;

4. PROPERTIES OF MODELING SERVICE SYSTEMS

Compared to the definition of hypergraphs, SSGs allow the existence of a predicate between two hypergraphs, which is represented by ψ_t. In order to model, we will present a selection of three modeling properties in the following section:

Multi-required (MR) service object: In an application environment, one service object is required by multiple activities, that is, it can be the input of more than one activities. According to the definition of SSG⟨R, A, Ψ⟩ and Ψ(Ψ^{-}, Ψ^{+}) we said a service object is a multi-required service object if $\bigcap_{i=1}^{n} \psi\, i(o) \neq \emptyset$ where $\psi \in \Psi^{-}$ and $\exists o \in O$, $n \geq 2$. In Figure 4 (a) the service object O3, O4 are multi-required service object for activities ψ2, ψ5 and ψ3, ψ4.

***Multi-delivered (MD) service object*:** One service object can be delivered by more than one activity. It is similar to or operators. According to the definition of SSG(R, A, Ψ) and tuple Ψ (Ψ-,Ψ+) a service object is called a multi-delivered service object when $\exists \bigcap_{i=1}^{n} \psi\, i(o) \neq \emptyset$, where $\psi \in \Psi+$ and $\exists o \in O$, $n \geq 2$. In Figure 4 (b) the service object O3, O4 are multi-required service object for activities ψ2, ψ5 and ψ3, ψ4.

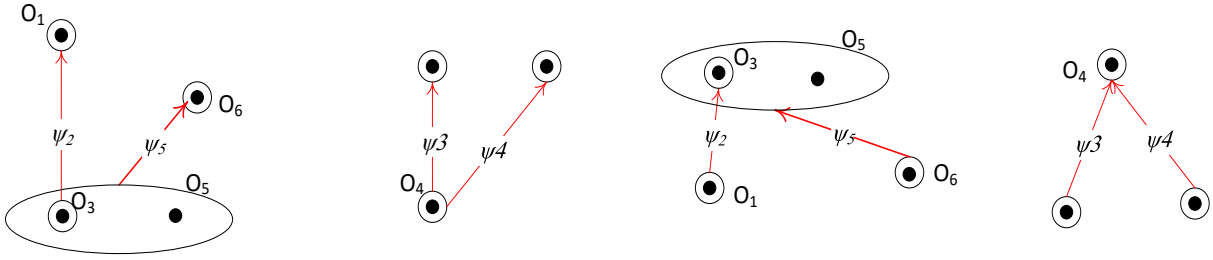


Figure 4: (a) Multiple Required Service Object and (b) Multi-delivered Service Object

***Sequence of service process*:** To model the service system, we still require to define the sequence of activities: Based on service system graph SSG(R, A, Ψ) and the service object O, subset of activities $\Psi_{after} \subset \Psi$ and $\Psi_{before} \subset \Psi$ with

$\Psi_{before} = \{\bigcup_{i=1}^{n} \psi_i \mid \bigcap_{i=1}^{n} \psi_i\,(o) \neq 0$ where $\psi \in \Psi+$ and $\exists\, o \in O\}$ and

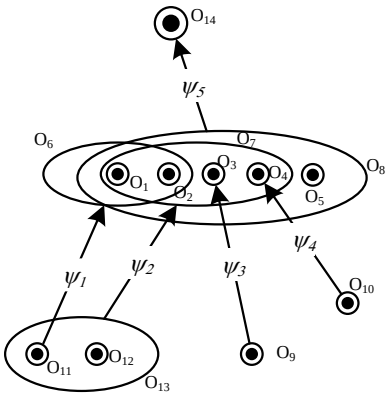
$\Psi_{after} = \{\bigcup_{i=1}^{n} \psi_i \mid \bigcap_{i=1}^{n} \psi_i\,(o) \neq 0$ where $\psi \in \Psi-$ and $\exists\, o \in O\}$, then $a \in \Psi_{after}$ follows each $b \in \Psi_{before}$.

We employ service system graph as a modeling approach to both formalize the relationships of configurations and visualize them using the inherent graphical notation. The next section discusses possible applications and areas of future research.

5. Application Scenario

Our modelling approach SSG has several application scenarios. The most evident one lies in its role as a tool to analyze both the organization's status quo and to structure possible alternative service system configurations. This chapter includes a detailed modeling example to show how this tool can be utilized. We focus on presenting both the graphical representation and the formal realization of a real-world service system scenario. The graphical illustration helps service systems engineers and business decision makers to structure their current business using a service systems perspective. It can also help authors make different system configurations of the same service apparent, thus giving decision makers the option to choose "paths" to reach their desired goal.

To illustrate our SSG, we modeled a possible service system based on our research project, an implementation of a CRM system at a mid-size German company "PowerCorp". PowerCorp faces the challenge of implementing a complex CRM system, for which they have tasked a team of IT consultants, service support providers, experts from the software provider and researchers. For the success of the IT-enabled organizational change project (Markus 2004), four core services have been commonly understood as crucial: First, the technical task of installing and configuring the CRM system based on the PowerCorp's existing IT-infrastructure. Second, the users require sufficient training using workshops or online courses that are specifically tailored to the needs of both the user's and the system's technical configurations. Third, the implementation of the CRM system requires extensive organizational analyses, which are usually provided by IT-consultants (including system tests). Fourth, the organization requires extensive after sales service support in case new requirements or questions arise. Part of the support requirements is also realized by a crowd support approach, which utilizes the potential of peer-advice hidden among PowerCorp's business units (Li et al. 2017).



O_1 System Integration and Setup
O_2 Training Support Service
O_3 Implementation & Testing Service
O_4 After Sales Service

O_5 Internal domain-knowledge service
O_6 Solution A: Basic Support Service
O_7 Total Solution Service
O_8 Turn-Key Service

O_9 Consulting Practices
O_{10} Hotline Service
O_{11} Solution A: Basic Support Practices
O_{12} Solution B: Implementation Practices
O_{13} Solution C: Total Solution Practices

ψ_1 Solution A: Configuration
ψ_2 Solution C: Configuration
ψ_3 Consulting Configuration
ψ_4 Hotline service Configuration
ψ_5 Turn Key service provisioning

O_{14} Project Success

Figure 5: Service system graph of the CRM implementation project

Figure 5 shows a service system with its service objects and activities. O_1-O_4 are the key service objects that are required for a successes project O_{14}. They cover the above-described core services. Each service object consists of resources and an actor. See below for a complete and detailed list of all elements.

For a successful CRM system implementation, an organization also requires involvement from business units not just an external project team consisting of consultants. By relying on key users, valuable contextual domain-knowledge can be integrated to overcome potential organizational pitfalls. We therefore define the service object O_5 as an internal domain-knowledge service.

All five service objects are necessary, while using a SSG helps in clearly denoting where they come from. We view O_1 and O_2 as basic support services, whereas a total solution would include all implementation and testing services, as well as ongoing after sales services (O_7). If these services are guaranteed and the necessary adaptations were made by incorporating rich domain-specific knowledge provided by O_5, the reins can finally be handed over in terms of a turn-key service (O_8) to the board of directors. The provisioning of the turn-key service (O_8) to project success is represented by ψ_5.

To understand how we reach O_7, we see them as MD service objects. This helps us realize that there are two possibilities in providing a total solution to PowerCorp: (A) By configuring total solution practices (O_{13}) to the contextual conditions ψ_2. (B) Configuring the basic support practices O_{11} accordingly ψ_1 and configuring both the hotline services O_9, ψ_3 and consulting portfolio O_{10}, ψ_4. It lies with the decision maker (e.g. service systems engineer) to choose which path to order to reach project success. As Figure 6 shows, the different service system configurations are very similar to “paths”, with two alternative paths highlighted in the graphic.

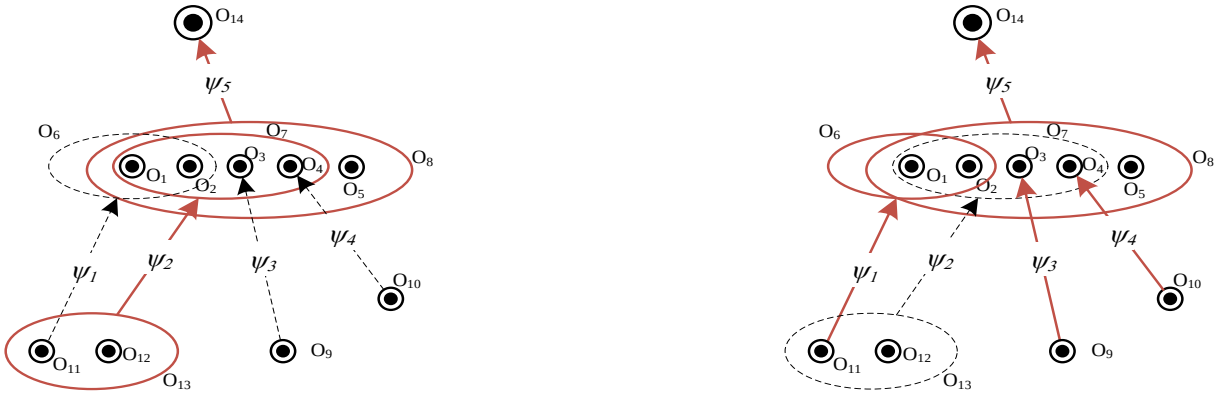


Figure 6: Service activities with optional path

The resulting service system is linked to a detailed list of resources and actors. This information is important for implementing a SSG as a software system. Relying on our formal definitions, we use can derive machine-readable data formats to integrate other systems. The following paragraphs give a detailed information on the service systems

structure and thus according to the application scenario, SSG(R, A, Ψ) is described as follows:

R={r_1: CRM Software, r_2: Hardware, r_3: Crowd Support System, r_4: Netware; r_5: Customer Specialist, r_6: Project Specialist; r_7: Software Trainer, r_8: Hardware Engineer; r_9: Software Developer r_{10}: System Analyst, r_{11}: Project Leader; r_{12}: Telephone Service Support, r_{13}: Server with OS; r_{14}: Documents, r_{15}: Tester; r_{16}: System Platform $r_{17:}$ Key Users; r_{18}: CRM Platform, r_{19}: Crosse Support/Online Platform, r_{20}: Running Application Hotline}

A=(a_1={r_1, r_2, r_3, r_4,r_7}:Package A definition, a_3={r_6, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15}}:Implementation Team, a_4={r_6, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15}, O_{11}}: Total Solution Concept with Team, a_5={O_1, O_2}: Solution A Team with Training, a_6={ Ò$_1$, O_2, O_3, O_4}: Project Solution Concept with Team, a_7={ O_7, O_5 }: Service Contract & Project Team, a_8={ r_6, r_9, r_{10}, r_{11}, r_{14}, r_{15} }: Business Analyze & Implementation Team, a_9={ r_{12}, r_{13} }: Project Team, a_{10}={ r_5 } Customer Project Team).

Ψ={ψ_1 Solution A Configuration, ψ_2 Solution C Configuration, ψ_3 Consulting Configuration, ψ_4 Hotline service Configuration, ψ_5 Turn Key service provisioning} with $\psi_1(O_{11})=O_6$; $\psi_2(O_{13})=O_7$; $\psi_3(O_9)=O_3$; $\psi_4(O_{10})=O_4$; $\psi_5(O_8)=O_{14}$}.

O={O_1(a_0,{ r_{16} }), O_2(a_0, {r_{17} }), O_3(a_0,{ r_{18} }), O_4(a_0, {r_{19} }), O_5(a_{10}, {r_5}), O_6(a_5, r_{16}, r_{17}), O_7(a_6, {r_{16}, r_{17}, r_{18}, r_{19}}), O_8(a_7, { r_{16}, r_{17}, r_{18}, r_{19}, r_5}), O_9(a_8, {r_6, r_9, r_{10}, r_{11}, r_{14}, r_{15}}), O_{10}(a_9, { r_{12}, r_{13} }), O_{11}(a_1,{ r_1, r_2, r_3, r_4, r_7 }), O_{12}(a_3, { r_6, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15} }), O_{13}(a_4,{r_1, r_2, r_3, r_4, r_6, r_7, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15} }), O_{14}(a_0, r_{20})}.

6. Discussion and Future Work

In conclusion, a SSG can be utilized to model complex service systems. Consider Figure 3, where O_5 is the service object for the end-customer a5. We see the equivalents of "OR" and "AND" operators. To create service object O3, one can either choose (ψ_2 AND ψ_4) or one can choose activity ψ_1. For the first, both actor a_2 and a4 are required, whereas a_2 utilizes shared resources from two different actors. As an alternative path to O_3, ψ_1 is also an option. This enables us to model different configurations of one service system as sub-systems, leveraging the systems of a systems principle (Bertalanffy 1972).

This enables service engineers to model their service systems both from a process perspective and from a structural perspective. Theoretically, we employed the input-output perspective for our modelling approach thus strengthening a service approach that does not differentiate between traditionally goose-dominant logic (Howard Lightfoot et al. 2013).

Furthermore, the formal description can be transferred into a corresponding database. This would enable an integration of our service system model with existing Enterprise Systems and applications. Similarly, the SSG approach would also greatly benefit from a set of computer-aided tools to model a service system based on our concept. Such a tool would greatly benefit from an interface to other databases.

Future SSG research should also consider focusing on manufacturing (Gallo and Scutellà 1998) and operations applications. SSG enables us to cross the divide of process and structural models, with the latter often including bill of materials (Hegge and Wortmann 1991). Since both approaches are based on simple graphs, a hypergraph based SSG enables the combination of both, whereas future research could focus on projections from SSG, which could be used to map the relation of traditional process or structural graphs to a SSG (Baget 2003).

To sum up, the paper presents the foundation of a hypergraph-based service system graph, which can be used to model service systems both formally, as well as graphically. Our concept grounds the concepts of a service systems using hypergraph theory and helps to demarcate the distinction between service systems and service ecosystems (Frost and Lyons 2017). Finally, future research could also include applying SSG into real-world scenarios and the limitation of our service system conceptualization through additional specifications.

References

[1] Alter S (2017) Service System Axioms that Accept Positive and Negative Outcomes and Impacts of Service Systems. International Conference on Information Systems 38:1–21.

[2] Baget J-F (2003) Simple Conceptual Graphs Revisited: Hypergraphs and Conjunctive Types for Efficient Projection Algorithms. International Conference on Conceptual Structures 2746:229–242.

[3] Berge C, ed. (1989) Hypergraph: Combinatorics of Finite Sets (Elsevier).

[4] Bertalanffy L von (1972) The History and Status of General Systems Theory. The Academy of Management Journal 15(4):407–426.

[5] Beverungen D, Lüttenberg H, Wolf V (2018) Recombinant Service Systems Engineering. Business Information Systems Engineering 21(1):50.

[6] Bitner MJ, Ostrom AL, Morgan FN (2007) Service Blueprinting: A Practical Technique for Service Innovation. California Management Review.

[7] Böhmann T, Leimeister JM, Möslein K (2014) Service Systems Engineering. Business Information Systems Engineering 6(2):73–79.

[8] Breidbach CF, Maglio PP (2015) A Service Science Perspective on the Role of ICT in Service Innovation. European Conference on Information Systems (ECIS) (AIS Electronic Library), http://aisel.aisnet.org/ecis2015_rip/33.

[9] Breidbach CF, Maglio PP (2016) Technology-enabled value co-creation: An empirical analysis of actors, resources, and practices. Industrial Marketing Management 56:73–85.

[10] Bretto A (2013) Hypergraph Theory (Springer International Publishing, Heidelberg).

[11] Frost R, Lyons K (2017) Service Systems Analysis Methods and Components: A Systematic Literature Review. Service Science 9(3):219–234.

[12] Gallo G, Scutellà MG (1998) Directed hypergraphs as a modelling paradigm. Decisions in Economics and Finance 21(1-2):97–123.

[13] Hegge HMH, Wortmann JC (1991) Generic bill-of-material: A new product model. International Journal of Production Economics 23(1-3):117–128.

[14] Hill TP (1977) On Goods and Services. Rev Income Wealth 23(4):315–338.

[15] Howard Lightfoot, Tim Baines, Palie Smart (2013) The servitization of manufacturing: A systematic literature review of interdependent trends. International Journal of Operations & Production Management 33(11/12):1408–1434.

[16] Leimeister JM (2012) Dienstleistungsengineering und -management (Springer Berlin Heidelberg, Berlin, Heidelberg).

[17] Li, M. M.; Peters, C. & Leimeister, J. M. (2017): Designing a peerbased support system to support shakedown. In: International Conference on Information Systems (ICIS). Seoul, South Korea.

[18] Lim C-H, Kim K-J (2014) Information Service Blueprint: A Service Blueprinting Framework for Information-Intensive Services. Service Science 6(4):296–312.

[19] Lusch RF, Nambisan S (2015) Service Innovation: A Service-Dominant Logic Perspective. Management Information Systems Quarterly (39):155–175.

[20] Maglio PP, Kieliszewski CA, Spohrer JC, eds. (2010) Handbook of Service Science (Springer US, Boston, MA).

[21] Maglio PP, Spohrer J (2008) Fundamentals of service science. Journal of the Academy of Marketing Science 36(1):18–20.

[22] Maglio PP, Spohrer J (2013) A service science perspective on business model innovation. Industrial Marketing Management 42(5):665–670.

[23] Maglio PP, Srinivasan S, Kreulen JT, Spohrer J (2006) Service systems, service scientists, SSME, and innovation. Communications of ACM 49(7):81.

[24] Patricio L, Fisk RP, Falcao e Cunha J, Constantine L (2011) Multilevel Service Design: From Customer Value Constellation to Service Experience Blueprinting. Journal of Service Research 14(2):180–200.

[25] Spohrer J, Maglio PP, Bailey J, Gruhl D (2007) Steps toward a science of service systems. Computer 40(1):71–77.

[26] van Eck P, Gordijn J, Wieringa R, eds. (2009) Value-Based Service Modeling and Design: Toward a Unified View of Services, 21st ed. (Springer, Berlin).

[27] Vargo SL, Akaka MA (2012) Value Cocreation and Service Systems (Re)Formation: A Service Ecosystems View. Service Science 4(3):207–217.

[28] Vargo SL, Lusch RF (2008) Service-dominant logic: Continuing the evolution. Journal of the Academy of Marketing Science 36(1):1–10.