# A Hypergraph-Based Modeling Approach for Service Systems

**Mahei Manhai Li, Christoph Peters and Jan Marco Leimeister**

**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.

**Keywords**  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 [27]. Since the seminal paper on service dominant logic [30] 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 [20]. Especially research on the transition to services has gained traction by terms, such as servitization [16].

M. M. Li (✉) · C. Peters · J. M. Leimeister
Information Systems, Research Centre for IS Design (ITeG),
University of Kassel, 34121 Kassel, Germany
e-mail: mahei.li@uni-kassel.de

C. Peters · J. M. Leimeister
Institute for Information Management, University of St.Gallen, 9000 St. Gallen, Switzerland

Service as a way of thinking has gradually evolved and is used by both manu-facturing and service-businesses, since production can be also be seen as internal services for providing an end-customer a value proposition [15]. Thinking in service systems can help identify service innovation potentials [5, 8]. However, we model service systems using a multitude of modeling approaches focusing on actors and a processual perspective [6, 19, 26, 28] or more technical perspectives. Yet, it can be challenging for practitioners to utilize the concept of service systems from a business perspective [1]. Since service scientists "study, manage, and engineer service sys-tems, solving problems and exploiting opportunities to create service innovations" [24], 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 [7, 16, 30]. Vargo and Lusch [30] have addressed the configuration in applying their concept of service dominant logic and called it resourcing. A service system is guided by a value proposition, which in turn has a corresponding configuration of actors and resources [7, 30].

We define a service system as a value co-creation configuration of resources [22]. This perspective is rooted in service dominant logic [23]. Resources include both operand and operant resources [21]. Different configurations of resources are connected by respective value propositions, sometimes also seen as service exchanges [29].

Recent research revisits the importance of value propositions and engagement of service systems [11], 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 [11, p. 1], whereas the creation of value ("value-in-use") happens through activities between actors, also referred as interactions [30]. This coincides with an input-output perspective, in which the realization of value happens through a transformation process of resources by actors [11].

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 imper-ative. 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 [3]. 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 [9]. 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 = $\{r_1, r_2, \ldots, r_n\}$;
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 $i \in \{1, 2, \ldots, n\}$.

Definition 1 shows that service object O is essentially a hypergraph [3]. 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 [10], 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.

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., [15]), 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 $i \in \{1, 2, \ldots, n\}$ [3]. 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 Fig. 1. 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.

**Fig. 1** Hypergraph G
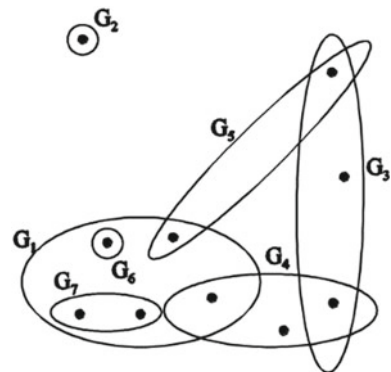
Although we have mapped service objects, which consist of actors and resources, we have yet to map a sequence of activities into our modeling approach. Up until now, a hypergraph can be used to model non-directional set of elements that contains the information of relationships among elements with the 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 [13]. 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 $\psi$. 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 $\psi(\psi^+, \psi^-)$

$$\text{with} \quad \psi : O : O \times O \to \text{Boolean} \quad \text{where} \quad O \times O \subset 2^O$$

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 $\psi$ is a tuple of $(\psi^+, \psi^-)$, which is a directed or counter-directed mapping of hypergraphs. In this paper, $\psi$ and $\psi^+$ are used synonymously for directed mapping (Fig. 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 [13].



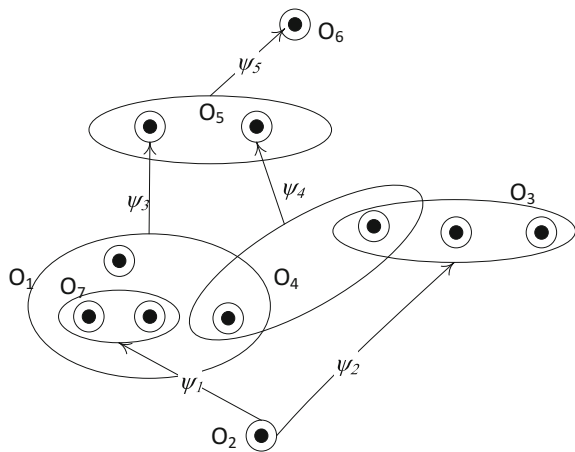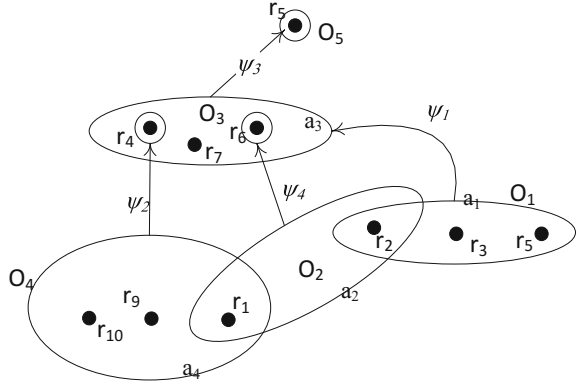**Fig. 2** Directed mapping of hypergraphs

**Fig. 3** Example service
system graph with $R_i \Leftrightarrow O_i$



**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 $\Psi$ be a set of functions as service activities. Then the tuple (R, A, $\Psi$) is called the SSG or service system graph,

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

Function $\Psi(O)$ defines which service objects are required as input factors and function $\Psi^+(O)$ 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 can include a configuration of service object and corresponding activities. This means that service systems can consist of service systems.

To illustrate the relationships of a service system, we present the detailed example of a SSG (R, A, $\Psi$): Fig. 3 shows the SSG with a set of resource $R = \{r_1, r_2, \ldots, r_{10}\}$, 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 $\psi_1 = ((a_1, \{r_2, r_3, r_5\}), (a_5, \{r_4, r_6, r_7\}))$; $\psi_2 = ((a_4, \{r_1, r_9, r_{10}\}), (a_0, \{r_4\}))$; $\psi_3 = ((a_5, \{r_4, r_6, r_7\}), (a_0, \{r_8\}))$; $\psi_4 = ((a_3, \{r_4, r_6, r_7\}), (a_0, \{r_5\}))$.

## 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 $\psi_t$. In order to model, we will present a selection of three modeling properties in the following section:
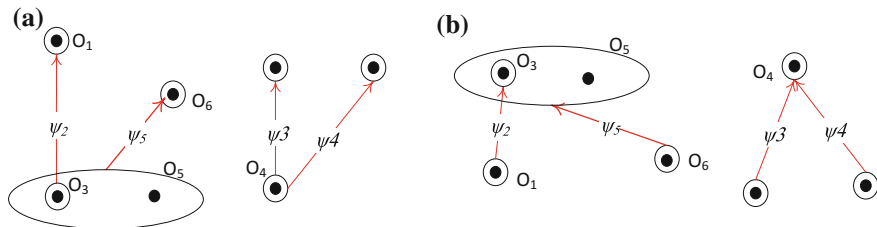
**Fig. 4 a** Multiple required service object and **b** multi-delivered service object

*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 $\langle R, A, \Psi \rangle$ and $\Psi(\Psi^-, \Psi^+)$ 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 Fig. 4a the service object $O_3$, $O_4$ are multi-required service object for activities $\psi_2$, $\psi_5$ and $\psi_3$, $\psi_4$.

*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, $\Psi$) and tuple $\Psi(\Psi^-, \Psi^+)$ 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 Fig. 4b the service object $O_3$, $O_4$ are multi-required service object for activities $\psi_2$, $\psi_5$ and $\psi_3$, $\psi_4$.

*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, $\Psi$) and the service object O, subset of activities $\Psi_{\text{after}} \subset \Psi$ and $\Psi_{\text{before}} \subset \Psi$ with

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

$$\Psi_{\text{after}} = \left\{ \bigcup_{i=1}^{n} \psi_i \mid \bigcap_{i=1}^{n} \psi_i(o) \neq 0 \text{ where } \psi \in \Psi^- \text{ and } \exists\, o \in O \right\},$$

then $a \in \Psi_{\text{after}}$ follows each $b \in \Psi_{\text{before}}$.

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 "Power-Corp". 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 [25], 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 [18].

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
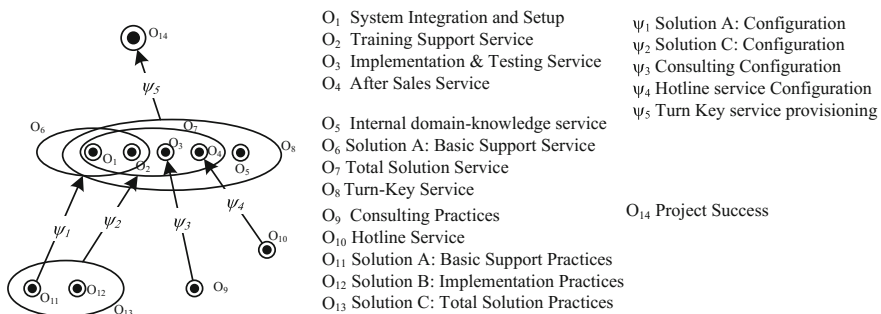


$O_1$ System Integration and Setup
$O_2$ Training Support Service
$O_3$ Implementation & Testing Service
$O_4$ After Sales Service

$O_5$ 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

$\psi_1$ Solution A: Configuration
$\psi_2$ Solution C: Configuration
$\psi_3$ Consulting Configuration
$\psi_4$ Hotline service Configuration
$\psi_5$ Turn Key service provisioning

$O_{14}$ Project Success

**Fig. 5** Service system graph of the CRM implementation project

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 $\psi_5$.

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 $\psi_2$. (B) Configuring the basic support practices $O_{11}$ accordingly $\psi_1$ and configuring both the hotline services $O_9$, $\psi_3$ and consulting portfolio $O_{10}$, $\psi_4$. It lies with the decision maker (e.g. service systems engineer) to choose which path to order to reach project success. As Fig. 6 shows, the different service system configurations are very similar to "paths", with two alternative paths highlighted in the graphic.

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, $\Psi$) 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 Plat-
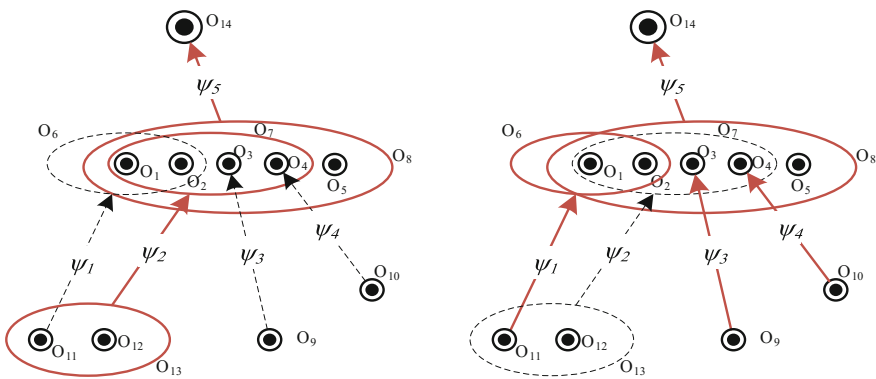


**Fig. 6** Service activities with optional path

form $r_{17}$: 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, $a_6 = \{O_1, O_2, O_3, O_4\}$: Project Solution Concept with Team, $a_7 = \{O_7, O_5\}$: Service Contract and Project Team, $a_8 = \{r_6, r_9, r_{10}, r_{11}, r_{14}, r_{15}\}$: Business Analyze and Implementation Team, $a_9 = \{r_{12}, r_{13}\}$: Project Team, $a_{10} = \{r_5\}$ Customer Project Team).

$\Psi = \{\psi_1$ Solution A Configuration, $\psi_2$ Solution C Configuration, $\psi_3$ Consulting Configuration, $\psi_4$ Hotline service Configuration, $\psi_5$ 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 Fig. 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 ($\psi_2$ AND $\psi_4$) or one can choose activity $\psi_1$. For the first, both actor $a_2$ and $a_4$ are required, whereas $a_2$ utilizes shared resources from two different actors. As an alternative path to $O_3$, $\psi_1$ 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 [4].

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 [16].

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 [13] and operations applications. SSG enables us to cross the divide of process and structural models, with the latter often including bill of materials [14]. 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 [2].

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 [12]. 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. Service system axioms that accept positive and negative outcomes and impacts of service systems. Int Conf Inf Syst. 2017;38:1–21.
2. Baget J-F. Simple conceptual graphs revisited: hypergraphs and conjunctive types for efficient projection algorithms. Int Conf on Concept Struct. 2003;2746:229–42.
3. Berge C (ed). Hypergraph: combinatorics of finite sets. Elsevier; 1989.
4. von Bertalanffy L. The history and status of general systems theory. Acad Manag J. 1972;15(4):407–26.
5. Beverungen D, Lüttenberg H, Wolf V. Recombinant service systems engineering. Bus Inf Syst Eng. 2018;21(1):50.
6. Bitner MJ, Ostrom AL, Morgan FN. Service blueprinting: a practical technique for service innovation. Calif Manag Rev. 2007.
7. Böhmann T, Leimeister JM, Möslein K. Service systems engineering. Bus Inf Syst Eng. 2014;6(2):73–9.
8. Breidbach CF, Maglio PP. A service science perspective on the role of ICT in service innovation. In: European conference on information systems (ECIS), AIS Electronic Library; 2015. http://aisel.aisnet.org/ecis2015_rip/33.
9. Breidbach CF, Maglio PP. Technology-enabled value co-creation: an empirical analysis of actors, resources, and practices. Ind Mark Manag. 2016;56:73–85.
10. Bretto A. Hypergraph theory. Heidelberg: Springer International Publishing; 2013.
11. Chandler JD, Lusch RF. Service systems: a broadened framework and research agenda on value propositions, engagement, and service experience. J Serv Res. 2015;18(1):6–22.
12. Frost R, Lyons K. Service systems analysis methods and components: a systematic literature review. Serv Sci. 2017;9(3):219–34.
13. Gallo G, Scutellà MG. Directed hypergraphs as a modelling paradigm. Decis Econ Finan. 1998;21(1–2):97–123.
14. Hegge HMH, Wortmann JC. Generic bill-of-material: a new product model. Int J Prod Econ. 1991;23(1–3):117–28.
15. Hill TP. On goods and services. Rev Income Wealth. 1977;23(4):315–38.
16. Lightfoot H, Baines T, Smart P. The servitization of manufacturing: a systematic literature review of interdependent trends. Int J Oper Prod Manag. 2013;33(11/12):1408–34.
17. Leimeister JM. Dienstleistungsengineering und -management. Berlin, Heidelberg: Springer; 2012.
18. Li MM, Peters C, Leimeister JM. Designing a peerbased support system to support shakedown. In: International conference on information systems (ICIS). South Korea: Seoul; 2017.
19. Lim C-H, Kim K-J. Information service blueprint: a service blueprinting framework for information-intensive services. Serv Sci. 2014;6(4):296–312.
20. Lusch RF, Nambisan S. Service innovation: a service-dominant logic perspective. Manag Inf Syst Q. 2015;39:155–75.
21. Maglio PP, Kieliszewski CA, Spohrer JC, editors. Handbook of service science. Boston, MA, US: Springer; 2010.

22. Maglio PP, Spohrer J. Fundamentals of service science. J Acad Mark Sci. 2008;36(1):18–20.
23. Maglio PP, Spohrer J. A service science perspective on business model innovation. Ind Mark Manag. 2013;42(5):665–70.
24. Maglio PP, Srinivasan S, Kreulen JT, Spohrer J. Service systems, service scientists, SSME, and innovation. Commun ACM. 2006;49(7):81.
25. Markus ML. Technochange management: using IT to drive organizational change. J. Inf. Technol. 2004;19(1):4–20.
26. Patricio L, Fisk RP, Falcão e Cunha J, Constantine L. Multilevel service design: from customer value constellation to service experience blueprinting. J Serv Res. 2011;14(2):180–200.
27. Spohrer J, Maglio PP, Bailey J, Gruhl D. Steps toward a science of service systems. Computer. 2007;40(1):71–7.
28. van Eck P, Gordijn J, Wieringa R, editors. Value-based service modeling and design: toward a unified view of services. 21st ed. Berlin: Springer; 2009.
29. Vargo SL, Akaka MA. Value cocreation and service systems (re)formation: a service ecosystems view. Serv Sci. 2012;4(3):207–17.
30. Vargo SL, Lusch RF. Service-dominant logic: continuing the evolution. J Acad Mark Sci. 2008;36(1):1–10.