Supply Chain as a Service: A Cloud Perspective on Supply Chain Systems

Joerg Leukel, Stefan Kirn, *Member IEEE*, and Thomas Schlegel, *Member IEEE*

***Abstract*—Supply chains are characterized by multiple firms providing their resources and processes to meeting customer demand in an efficient manner. There exists a great variety of approaches for solving the inherent coordination problem. It has been acknowledged that the autonomy of supply chain participants – and their business objectives – as well as the contractual relations between participants have to be maintained when designing a coordination mechanism. Often, new mechanisms are triggered by innovation in the software industry. This paper adopts the basic idea of Cloud Computing and takes a Cloud perspective on supply chain systems: It proposes to represent supply chains as a set of service offerings and customer demand as service requests; coordination is then a problem of determining optimal service compositions. We evaluate our proposal in a case study of airport service supply chains.**

I. Introduction

In many industries, supply chains are challenged by their increasing complexity and the need for higher flexibility to meeting individual customer requirements (e.g., automotive, electronics, engineering, construction) [1] [2]. Where suppliers or customers on one hand break away (supply chain disintegration, disintermediation), on the other hand new partners offering value-added services for OEMs and customers need to be integrated. Business relations change as well: from long-term contracts (static supply chain systems) to small short-term contracts and even to event-driven management for dynamically creating and delivering individualized products.

At the same time, recent developments in software technology address a very similar problem in computing: How to fulfill individual and complex user requirements by effectively and efficiently using distributed resources? This question has led to the idea of Cloud Computing [3]. A Cloud is basically a pool of shared resources. Users do not need to have detailed knowledge about available resources and their properties, but search for services, consume resources as a service, and then pay only for the services used. A middleware provides respective functionality for matching demand and supply.

The objective of our research is to develop a new approach for supply chain coordination by adopting the idea of Cloud Computing: We analyze supply chain systems from the perspective of Cloud Computing and thus propose 'Supply Chain as a Service' (SCaaS). Unlike computational Clouds, we represent supply chain operations such as transportation, warehousing etc. as software-based services. Thus we do not represent supply chain IT functionality such as tour planning, but use the Cloud idea for a new class of software-based services. The designation SCaaS must be understood as a metaphor for giving access to operations within a supply chain by means of standardized electronic interfaces.

The approach consists of two main elements: (1) representation means for supply chain operations and (2) coordination means for supply chain services. We evaluate the proposal by reporting about experiences made in adopting and implementing its elements. The use case concerns a supply chain of ground handling operations at airports.

The rest of the paper is organized as follows. Section 2 introduces the Cloud perspective on supply chain systems. Section 3 proposes a model for describing supply chain services. Section 4 extends this model for coordination of supply chain services. In section 5, we provide a preliminary evaluation and discussion. Section 6 discusses related work. Finally, we draw conclusions and outline avenues of future research.

This is an Accepted Manuscript of an article published by IEEE in
IEEE Systems Journal on 2011-02-18, available online: https://doi.org/10.1109/JSYST.2010.2100197

To cite this article:
Leukel, J., Kirn, S., & Schlegel, T. (2011). Supply chain as a service: A cloud perspective on supply chain systems. *IEEE Systems Journal*, 5(1), 16-27, DOI: 10.1109/JSYST.2010.2100197.

II. Cloud Perspective

In this section, we introduce the basic concepts of Cloud Computing and map these to supply chain systems.

A. Cloud Definition

The term Cloud Computing emerged in 2007. It describes a concept for the provision of resources. By virtualization, resources can be easily used in the form of services. Service providers make their services available to service users by Internet-based interfaces. Depending on the resource type and the provided capability, the Cloud concept enables three scenarios: (1) Infrastructure as a service (IaaS), describing pure computation resources such as storing and processing capacity, (2) platform as a service (PaaS), describing software platforms, i.e., machines with an operation system, and (3) software as a service (SaaS), describing software applications that run in the Cloud.

The concept has attracted a lot of attention, driven by early commercial implementations such as Amazon Elastic Computing Cloud (EC2) and Google App Engine. There still exists no commonly accepted and solid definition for Cloud. Because it describes a complex idea, it is associated with a variety of similar technologies such as Grid Computing and Service-oriented Computing (SOC). In addition, the recent hype has made it difficult to identify its key and distinguishing features. In the following, we ground the Cloud perspective on the definition proposed by Vaquero et al. [4], who analyzed twenty-two definitions. This definition shows how the concept is perceived today.

Definition 1 (Cloud): A *Cloud* is "a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized Service Level Agreements (SLA)" [4].

B. Rationale for Supply Chain Systems

Describing supply chain systems through the lens of Cloud Computing, we first need to define *resource type*: Supply chain systems are constituted by resources that are involved in delivering goods (and services) from a supplier/source to a customer [5]. Computational resources are used in conjunction to enable and facilitate dynamics and flexibility of supply chain systems. These resources span a wide range, e.g., including manufacturing sites, warehouses, transportation vehicles, employees, and their properties and interrelations. Following the Cloud concept, this variety and complexity, however, can be greatly reduced by *virtualization*: It hides the heterogeneity of the underlying resources by a predefined interface.

The interface is used for accessing a *service* without determining a concrete resource instance to be used. In other words, a service represents the capability of a resource type for which typically more than one resource is available. Since we add a new type of services which does not exist in Cloud Computing, this scenario can be adequately denominated as Supply Chain as a Service (SCaaS).

Cloud Computing assumes a *large set* of resources of the same resource type; the ultimate objective is to fulfill all user demand, i.e., service requests, while at the same time maximizing utilization across resources. In supply chain systems, this is similar to a key objective of supply chain management (SCM): lowering the costs required to provide the necessary level of customer service to a specific segment [6]. The 'large set' property emphasizes the existence of alternatives for delivering a requested service (e.g., choosing the cheapest resource). In supply chain systems, the picture is very much the same for commodities; though other elements of a supply chain system may not be easily (or not at all) replaced due to nonexistence, exclusion of competition, or other restrictions.

Reconfiguration is the process of adding, removing, or modifying resources depending on the customer demand. It is closely associated with scalability; the Cloud is a dynamic set of resources depending on the current and forecasted load. Supply chain systems also rely on scalability, e.g., by contracting new suppliers to meet unexpected demand or shifting jobs due to resource failures. Scalability is, however, lower compared to computing. For instance, constraints increase the time required for reconfiguration (e.g., physical distances, technical and organizational interfaces). Scalability of SCaaS is enabled by outsourcing supply chain functions via information systems; the underlying theory is transaction cost economics and its impact on interorganizational coordination [7]. These effects hold also for Web services computing and provide respective opportunities for supply chain systems.

The *pay-per-use model* is made possible by the transformation of computing into a utility, such as water, power, and transportation. Such services can easily be metered and thus charged based on actual consumption. The Cloud did not first introduce this concept, but incorporated the utility computing model [8]. Its adoption to supply chain systems depends largely on the commodity property of services. Commoditization describes a situation in which competitors offer increasingly homogenous goods or services to price-sensitive customers. It is being observed in many industries [9] and a driver for outsourcing business processes [10]. Commoditization helps turning fixed costs into variable costs, but may also erode the competitive differentiation of firms [11]. From this perspective, Cloud Computing can be seen

as a technology for materializing commoditization into electronic business processes. Evaluating its potential, however, has to consider the organizational setting and cultural determinants, which are out of the scope of this work.

With regard to its organization, a Cloud typically distinguishes two roles, *Infrastructure Provider* (IP) and *Service Provider* (SP). Whereas the IP provides the actual resources, the SP delivers services using such resources. Ideally, the two roles are fulfilled by different organizations; thus the SP can concentrate its efforts on services without catering for all implementation issues. Therefore, only the SP interacts with users. The obligations and guarantees regarding services in the business relationship between IP and SP are defined explicitly in an SLA [12]. An important language for formally specifying SLAs is WS-Agreement [13].

In supply chain systems, the distinction of IP and SP can be mapped to any subsystem of a firm (SP) serving a customer and one or more upstream tiers (IP). Inside these upstream tiers, contracts exist between supply chain participants and thus find a counterpart in SLAs. From the customer's perspective, only the interface, thus the SP, is of relevance, whereas the SP is in charge of managing its supply chain (set of IPs). In this sense, the customer perceives the supply chain system as a black-box. The term Cloud comes very close to this metaphor, with users delegating the solution how to provide the requested service to a vague 'Cloud'.

C. Role of Service Level Agreements

In a Cloud, *SLAs* are used primarily for managing the resource provision inside the Cloud. It has been criticized that Clouds do not guarantee certain service levels to the user [4]; this limitation needs to be lifted for supply chain systems, due to the importance of customers. Supply chain systems show a higher diversity of exchange relationships between IP, SP, and the customer than Clouds. In particular, these relationships are subject to power regimes [14], being the result of strategies in procurement respectively marketing/distribution. In case of buyer dominance, an SLA between SP and customer would affect or even determine also SLA parameters of upstream SLAs. In case of supplier dominance, a weak upstream SLA performance would severely limit the SLA performance of the entire supply chain system, and thus affect the SLA between SP and customer. In an extreme situation, the weakest SLA performance would become the SLA for the supply chain system.

In Cloud computing, interdependencies between SLAs can be formally represented by the concept of QoS (Quality of Service) aggregation respectively disaggregation [15]. It provides mechanisms to determine respectively break-down SLAs while considering the service system, e.g., sequences, parallel services, as well as different SLA parameters such as throughput, response time, reliability etc.

D. Preliminary Summarization

Table 1 summarizes the findings of analyzing supply chain systems from a Cloud Computing point of view.

TABLE 1
CLOUD PERSPECTIVE ON SUPPLY CHAIN SYSTEMS

Element	**Cloud Computing**	**Supply Chain Systems**
Resource type	Hardware, machines, software applications	Physical and human resources in supply chain systems
Service type	Infrastructure (IaaS), Platform (PaaS), and Software as a Service (SaaS)	Supply Chain as a Service (SCaaS)
Number and specificity of resources	High number; specificity relatively low due to IT standardization	High number; specificity low due to commoditization, but also high due to differentiation as strategic alternatives
Reconfiguration	Adding/removing resources results in high scalability; quick adaptation to demand volatility (e.g., within seconds)	Physical, technical and organizational constraints limit and slow down adding/removing resources.
Pay-per-use model	Based on standard measurements such as processing time, data volume	Supported by commoditization; most often based on actual load combined with performance metrics
Infrastructure Provider (IP)	Provides actual resources	Represents upstream tier of any supply chain participant
Service Provider (SP)	Provides services to the final customer by contracting IPs	Represents downstream tier, ultimately the one serving the final customer
Service Level Agreements	Exist between IP and SP only	Exist for any contractual relationship in the supply chain system, including the final customer

Most elements can be mapped to respective phenomenon. Literature yields evidence of properties such as commoditization and customer-orientation that support this perspective. However, the analysis also points to a greater *diversity* and *richness* of resources which has to be addressed in service description and composition; it increases the interoperability problem in the Cloud. Additionally, supply chain systems are made of autonomous entities which aim at different, overarching and conflicting goals. Acknowledging this characteristic, coordination mechanisms for supply chain services need to be carefully selected and designed.

III. SUPPLY CHAIN SERVICE MODEL

This section proposes a formal model for describing supply chain services. Following the Cloud perspective, services are of utmost importance. When we use the term 'service' for supply chain systems, we refer to a software-based representation, but not the actual real-world service. In technical terms, such services are Web services. A Web service is a software system supporting interoperable machine-to-machine interaction over an electronic network [16]. The service is formally described in an interface. Interaction takes place by exchanging messages between provider and requester.

SCM practice and research has contributed a variety of modeling approaches for supply chains, ranging from quantitative decision models to semi-formal models supporting communication about and design of supply chains. These models do not address supply chain services explicitly, but include at least participants/actors as the most abstract type of resource and flows of goods. The latter can be used for arriving at services. We thus first define supply chain system and then service.

Definition 2 (Supply Chain System): A *supply chain system* consists of nodes participating in producing, transforming and/or moving a good or service from suppliers to customers [5]. The interrelations between nodes are constituted by the possible flows of goods (or service; note: real-world services, not computing services), whereas nodes represent the storage and/or production of goods (or services) at locations. Supply chain system is defined as a directed graph $SC=(N, F)$, where N is the set of all nodes and F is the set of all possible flows of goods with $F \subseteq N \times N \times G \times TM$. Each f is a 4-tuple $f=(n_j, n_k, g, tm)$, with flow of good g from n_j to n_k using the transportation mean $tm \in TM$.

The following integrity constraints must hold:

1. Let $\bullet n=\{m|(m, n) \in F\}$, the set of input nodes of n; then at least one $n \in N$ exists with $|\bullet n|=0$. Thus, at least one node has no incoming flows, i.e., it represents a real source of goods.
2. Let $n\bullet=\{m|(n, m) \in F\}$, the set of output nodes of n; then at least one $n \in N$ exists with $|n\bullet|=0$. Thus, at least one node has no outgoing flows, i.e., it represents a final destination of goods.
3. For all $n \in N: |n\bullet|+|\bullet n| \geq 1$; the graph SC is (weakly) connected.

Definition 3 (Supply Chain Service): A *supply chain service* is a possible service flow from a supply chain service provider to a supply chain service consumer. The set of all such services form the *supply chain service flow model*, which is a directed graph $SF=(A, S, M)$. A is the set of all actors. S is the set of all possible elementary service flows. Each s is a tuple $s=(a_j, a_k)$, with flow s from a_j to a_k. M is a relation which maps each s to f in SC. Thus each service flow s can implement $|M_s|$ flows of goods in L.

The following integrity constraints must hold:

1. Let $\bullet a=\{b|(b, a) \in S\}$, then at least one $a \in A$ exists with $|\bullet a|=0$, i.e., actor which is not a service consumer.
2. Let $a\bullet=\{b|(a, b) \in S\}$, then at least one $a \in A$ exists with $|a\bullet|=0$, i.e., actor which is not a service provider.
3. SF is (weakly) connected.
4. For any s, if $|M_s| \geq 2$, then $t:= |M_s|$ and there must exist a walk w_s in SC with $ws=(n_i, m_{s,1}, .., m_{s,t}, n_j)$ and $n_i n_j \in N$. If a service implements two or more flows, then these flows form a walk from node n_i to node n_j. If w_s exists, then p_s is the path for w_s.

The last integrity constraint allows for defining a service that realizes the flow of goods between nodes that are not directly linked; for instance, delivering goods from a manufacturer via a distribution center to retail stores. However, the problem is that this service would be provided by an actor without unfolding whether the actor implements all flows of goods by himself or subcontracts other actors/service providers. This problem can be solved by adopting the concept of *service composition* of Web service research [17] [18]. A service composition is a combination of services; it has two dimensions: hierarchy, i.e., part-of relationships between services, and sequence, i.e., logical order of services.

Definition 4 (Composite Supply Chain Service): A *composite supply chain service* is a composition of two or more supply chain services in SF. It is defined as a 4-tuple $cs=(a_j, a_k, CE, WF)$. CE denotes the composition elements. WF denotes the workflow, i.e., logical sequence of all $ce \in CE$. The composite service is offered by actor a_j to actor a_k.

The following integrity constraints must hold:

1. For each composite service cs_i: $CE \subseteq S^* \cup CS \backslash cs_i$. The composition may include elementary and composite services (without the respective cs_i).
2. WF describes a directed graph (A', S', a_1) with edges $A' \subseteq A$, arcs $S' \subseteq S$, and root a_1.

Composition is another mean for abstracting: it abstracts from its included services by providing a composite interface for accessing all the contained services. More precisely, we should say for accessing the contained services according to a given workflow. In SOC and Cloud Computing, a composite service and its interface is offered to users; in supply chain systems, users would be called final customers. Therefore, we employ the idea of composite services for providing an interface to final customers of a supply chain (e.g., individuals in case of consumer products). These customers search for and request a solution for their problem (e.g., delivery of certain products including location, quantity, time, terms of delivery, etc.). Fulfilling this request is only possible by composing a respective service, based

on both customer requirements (demand) and a set of available supply chain services (supply). These supply chain services are abstractions from an underlying supply chain system of real-world resources. Fig. 1 shows the interrelations between the three formalisms that we have introduced so far. The final customer is represented as a_f for whom a composite service is offered by a_{cs}.

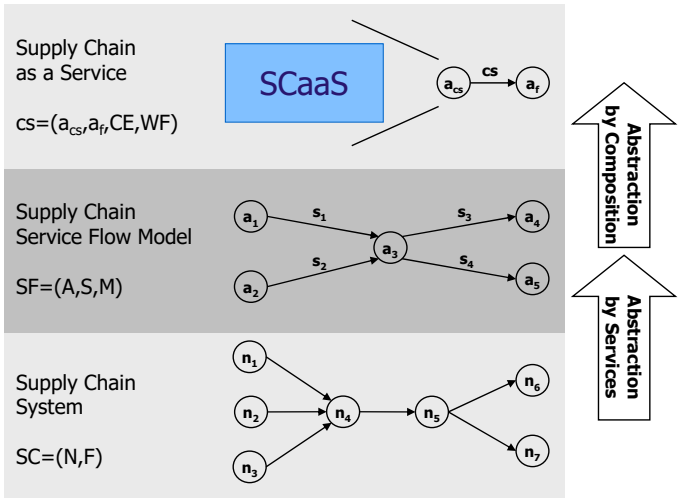

Fig. 1. SCaaS is a two-step abstraction from supply chain systems.

IV. Coordination of Supply Chain Services

In this section, we extend the proposed supply chain service model by means for describing relationships between services. We first identify levels of supply chain service coordination, and then propose respective models.

A. Coordination Issues

A service-oriented representation of supply chain systems inherits the coordination problem of supply chains. Following the Cloud perspective, we propose adopting service composition from SOC. An essential requirement of composition is interoperability of services. Before coordination mechanisms can be designed, interoperable service descriptions have to be established. Interoperability in SOC is being addressed by the idea of *Semantic Web Services* which provide not only a syntactical but semantic description of service interfaces. These interfaces have a functional description including input and output data and other attributes required for capturing the conceptualization of the service. An example model for such interfaces is OWL-S (Semantic Markup for Web Services) which defines a standard set of attributes [19].

Arriving at a unified semantic service description is, however, only a first step: We also need to analyze the data flows between services. More specifically, we need to map output data to input data of services. These mappings can then be used for exchanging and transforming instance data during the actual interaction of two services. In summary, we can identify several levels of supply chain service coordination, as shown in Fig. 2.

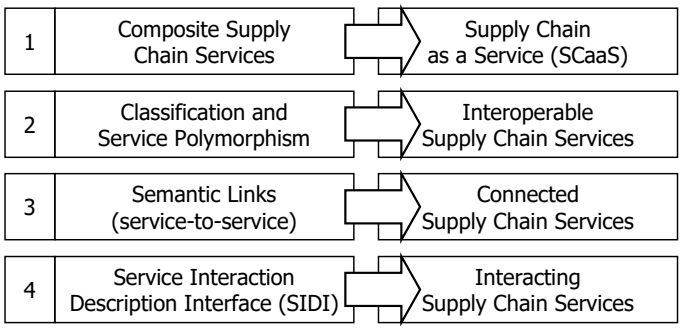

Fig. 2. Levels of supply chain service coordination.

The four levels are interrelated as follows:

1. On the top-level, we use the idea of composite services for enabling SCaaS.
2. On the second level, these services are classified to become interoperable (see subsection B and C).

3. On the third level, services are connected by Semantic Links which define the relation between output of the preceding and input of the proceeding service (see subsection D). The concept of Semantic Links originates from Semantic Web services **Fehler! Verweisquelle konnte nicht gefunden werden.**.
4. On the fourth level, service interaction description interfaces (SIDI) are used for transforming instance data, matching ambiguous data, and amending missing data interactively (see subsection E).

B. Service Classification

Definition 5: A *classification scheme* over a set of supply chain services *S2CS* is defined as $C=(P, p_0, V)$. P denotes the service types $p \in P$. V denotes inheritance relations in P. The root service type is denoted as p_0 from which all other service types are derived.

A candidate classification for supply chain services could be the Supply-Chain Operations Reference Model (SCOR) [21]. It provides a three-level classification of so called process types (e.g., source, make, deliver, plan, return), process categories (differentiated by customer specificity), and process elements (differentiated by logical sequence).

By classification, the graph model of supply chain services is extended with object-oriented modeling. The service description is then a triple $s_t=(a_1, a_2, p)$ with $p \in P$. An inheritance relation $v \in V \subseteq P \times P$ denotes a directed relation from a more specialized service type p_1 to the more general service type p_2, using single inheritance as basic concept and ensuring that each $p \in P$ is part of the inheritance tree; hence $p_1 \in P \setminus \{p_0\}$, $p_2 \in P$, $p_3 \in P$, $p_1 \neq p_2$, $p_2 \neq p_3$, $\iota\, p_1 \varphi v : v=(p_1, p_2) \omega \kappa v=(p_1, p_3)$.

The concept of transitive inheritance and classification is used to determine, if a specific service type p_u is allowed when a more general (or trivially the same) type p_r is requested by the workflow. This *service polymorphism* allows to define a supply chain service framework (providing all general service types) whereas it is still possible for service providers to define their own specialized services within the framework. For example, the framework would specify a composite service of type p_r (e.g., by referring to a SCOR process category). The service provider can then choose to internally implement a service of type p_u which follows the defined service type p_r but specializes and adds steps specific to the company (e.g., by adding SCOR process elements). The overall service instance of p_u will still be compatible with the required p_r for the supply chain service requested.

To ensure this semantic compatibility, we define the *super-type set*, which includes all types in the inheritance-based type hierarchy for a specific type p_u. It is defined recursively as a set:

$P_u = \{ p_u \}, p_u \in P$.

P_u contains all more general service flow types for p_u:

$P_u := P_u \cup \{p \in P \mid \varphi v=(p_r, p) \in V_p, p_r \in P_u \}$

$\iota\, p_r \in P_u$: p_u can polymorphically substitute p_r, i.e., p_u can be used everywhere where a p_r is requested (service polymorphism). The same concept applies to all types of elements e and their super-type set $E_u(e)$ in our approach and will be applied to input, inner and output data of a service type as described in the following. We adopted this concept from [22] which provides a more general model for classifying processes based on object-orientation and its inherent polymorphism.

C. Service Data

The current classification of supply chain services is not sufficient for interoperability, because it neglects the existence of data being input or output of services and therefore transferred between services. This aspect is important especially for services with different stakeholders. We include this data into the classification by joining the two sets of service types (P) and data types (D) to the set of elements (E): $E = P \cup D$ to include all active (P) and passive (D) elements.

The flow connection graph $G=(E, R)$ connects elements $e \in E$ using relations $r \in R \subseteq E \times E$.

$R = F \cup T \cup (V \cup V_p)$ consists of all flows F, aggregations T and inheritance relations V. It connects services and data to form an integrated model with services connected and ordered via F, aggregated to composite services with T and typed by the inheritance relations V.

The inheritance relations $v \in V$ between elements $e \in E$ in the model can connect each element only to one upper element and are therefore defined as follows:

1. $e_1 \in E \setminus \{p_0\}$, $e_2 \in E$, $e_3 \in E$, $e_2 \neq e_3$, $e_1 \neq e_2$, $v \in V$
   $\iota\, e_1 \varphi v : v=(e_1, e_2) \omega \kappa v=(e_1, e_3)$.
2. Cyclic inheritance graphs are not allowed, i.e.,
   $\iota\, e \in E\ \kappa v=(e_1, e)\ e_1 \in E_u(e)$.
3. Each service flow type p can contain a set of subcomponents $E_p \subseteq E$ using aggregation t: $\iota\, e_p \in E_p \varphi t \in T: a=(e_p, p)$.

Whereas elements are contained via aggregation, internal flow relations F_p of a service type p are defined as all flow relations that connect elements within a specific service type p creating a horizontal order and causality between the elements:

$\iota f_p \ni F_p \varphi p_1, p_2 \ni E_p: f_p = (p_1, p_2)$

Input elements e_i of a service type p are data elements $e_i \ni D_{p,i} \setminus (E_p \backslash P)$ that do not have an incoming internal flow connection but connect to internal components:

$\iota e_i \ni D_{p,i} \setminus (E_p \backslash P) \kappa f_p = (e_{p,1}, e_i) \ni F_p \omega \varphi f_p = (e_i, e_{p,2}) \ni F_p, e_{p,1}, e_{p,2} \ni E_p$.

On the other hand, output elements e_o of p are data elements that do not have an outgoing internal flow connection

$\iota e_o \ni D_{p,o} \setminus (E_p \backslash P) \kappa f_p = (e_o, e_{p,1}) \ni F_p \omega \varphi f_p = (e_{p,2}, e_o) \ni F_p, e_{p,1}, e_{p,2} \ni E_p$.

This implies that internal data elements e_m have both incoming and outgoing internal flow relations:

$\iota e_m \ni D_{p,m} \setminus (E_p \backslash P) (\varphi f_p = (e_m, e_{p,1}) \ni F_p \omega \varphi f_p = (e_{p,2}, e_m) \ni F_p), e_{p,1}, e_{p,2} \ni E_p$.

In a consistent model, there are no data elements of p that have neither input nor output bindings to any internal element, meaning that an input or an output binding has to exist in every component e_p.

$\iota e_p \ni D_p \setminus (E_p \backslash P) (\varphi f_p = (e_m, e_{p,1}) \ni F_p \varpi \varphi f_p = (e_{p,2}, e_m) \ni F_p), e_{p,1}, e_{p,2} \ni E_p$.

The distinction between input, inner, and output data extends the supply chain service model as shown in Fig. 3.

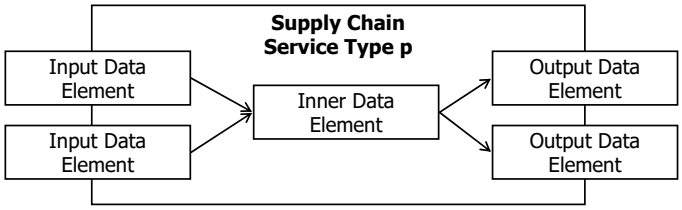

Fig. 3. Data elements of supply chain service types.

D. Service-to-Service Interface Using Semantic Links

A supply chain system has to transfer goods from one supply chain tier to the next one. Accordingly, a service-based representation has to assure correct transfer of data from one service to the next service. This service interface has to be considered when describing services and service compositions. It concerns the transfer of output data of the preceding service into input data of the proceeding service.

We consider a composite service *g0* as shown in Fig. 4. It consists of two subservices *g1* and *g2*; the service is provided from a provider a_1 to a final customer a_f .

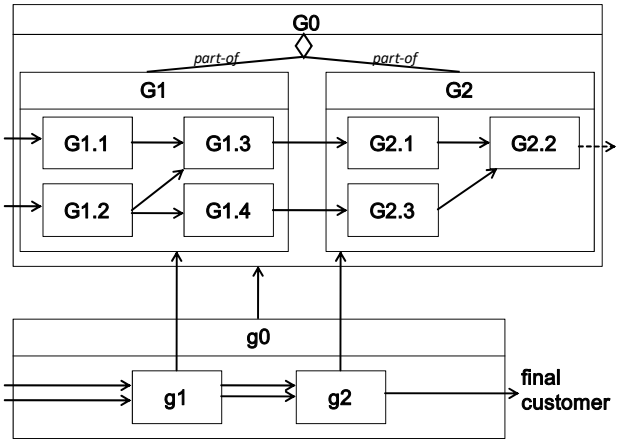

Fig. 4. Data flow in a composite service.

During execution of *g0*, service *g1* hands over data to service *g2*. Therefore, two service offerings may only be connected within a composite service, if *g1* has the right output data elements matching the required input data elements of *g2*. For specifying these requirements, we define a service-to-service Interface using *Semantic Links* [19]. The Semantic Links describe the interconnection of services on the *type* level, not on the instance level, i.e., *G0* is a composition of *G1* and *G2* with data flows *G1.3*→*G2.1* and *G1.4*→*G2.3* between *G1* and *G2*.

Following the object-oriented definition of data elements introduced before, every data input element can consume a data output element of the same type, or a more specialized type. This also implies that an output element may deliver the same or a specialized type that will be compatible. For instance, if a more general data element such as vehicle type ‘Truck’ is required, all output types can be used that contain the input type requested in their super-type set, e.g., ‘HeavyTruck’ or ‘ActrosTruck’.

E. From Data Mappings to Interactive Transformations

Once supply chain service composition has taken place and instances s_t are being executed, the actual data flows have to be realized. To provide the necessary data for properly executing the next service in the workflow, all requirements defined by the Semantic Links have to be fulfilled. This is achieved by filling all input data elements with instance data. We need to consider three cases:

1. *The necessary data is available from a preceding service and can be identified and matched automatically:* Matching and data transmission can either be accomplished type-based (unambiguous, i.e., types have to be different or must be matched manually – see 2) or relation-based (i.e., flow relations like *G1.3→G2.1* exist between input of the current service and output of the predecessor service). Relations and types in the meta-model define the handshake of Deliver and Source.
2. *The necessary data is available from a preceding service but matching is ambiguous:* Because of similar types it has to be achieved either by predefinition (operator-based) or by ad-hoc selection by the executor. Matching is supported by providing a semantically matching subset of data elements for selection, which is created to support the user in selecting. Semi-automatic semantic matching is possible by typological correctness being checked by the system.
3. *The necessary data from a preceding service is missing:* it has to be entered or selected ad-hoc by the executor. The set for selecting and editing is created using the type semantics and input types are controlled for match.

F. Service Interaction Description Interface (SIDI)

In the third case of not having necessary input data available as output data of a supply chain service being part of a composite service, it is necessary to provide runtime support for interactive and ad-hoc completion of necessary data. This interface is called *Service Interaction Description Interface (SIDI)*. It represents the final model layer for supply chain coordination by connecting services (see Fig. 5).

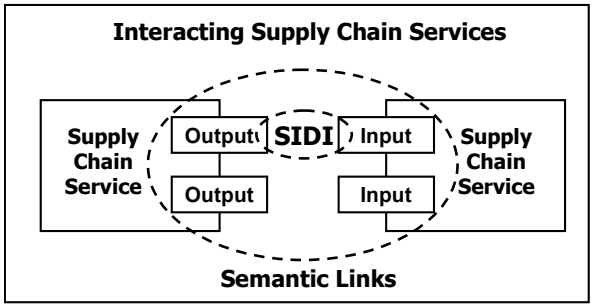

Fig. 5. SIDI connects supply chain services on the instance level.

For example, when a_1 hands over goods and (meta-)data to a_2, the system has to offer a_1 as well as a_2 the possibility to select or enter information necessary to complete the handover process (e.g., use of mobile device by staff responsible for supply chain service delivery).

When two supply chain services have been connected, a user interface enables stakeholders of the physical service process to realize not only the physical handover of goods but also to ensure correct supply chain service metadata being transferred.

For example, a_2 requires the names of all, i.e., including a_1's, drivers to be filled in the quality assurance form. The process of a_2 therefore contains the specific input element for this information. As the information is not present and therefore not automatically filled in by semantic matching, it will be shown on a_2's display as missing data and on a_1's display as requested data. Both a_1 and a_2 are then asked to complete the missing information – a_1 by selecting it from a list of "driver name" values or a_2 by entering the name manually, if the value is not provided by a_1.

This way, the model can help identifying input data components and output data components and show them to the user – including their types and data payload.

V. Evaluation

In this section, we provide a preliminary evaluation of our approach. We apply the proposal to a supply chain system of ground handling operations at airports. We describe the supply chain system, instantiate the proposed model elements, and report results of a prototype implementation. The purpose of the experiment is to demonstrate the feasibility and usefulness of the proposed models.

A. Supply Chain System

We consider ground handling at airports: It involves (1) ground handling resources, (2) ground handling companies, and (3) airlines. Airlines have contracts with ground handling companies about handling inbound and outbound flights. Ground handling companies own resources for executing the handling process. There are resources which offer more than one operation (e.g., staff for loading and unloading).

Whereas ground handling concerns physical items (e.g., aircrafts, baggage, passengers, etc.), it has to be noted that the handling process is being executed by several ground handling companies; each company provides handling operations (not goods). This fact makes it very simple to construct the supply chain system *SC* and supply chain service flow model *SF*. Actually, the latter's relation *M* is a bijection. An example *SF* is shown in Fig. 6 (limited to one ground handling company and airline to reduce presentation complexity).

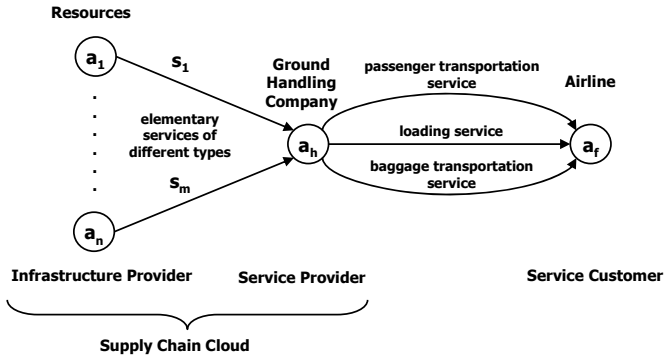

Fig. 6. Supply chain service flow model *SF* with *n* resources and *m* services; *h* denotes handling company, and *f* denotes final customer.

This at first sight static supply chain system is, however, affected by a highly volatile resource usage over time (peak-times in the morning and afternoon) and a very dynamic environment (flight delays due to internal and external plan deviations). There is a critical need for reacting to such changes in short-time, in particular for solving unforeseen bottlenecks, even during its execution. These characteristics motivate a 'servitization' which opens-up the static supply chain system and adopts service composition for problem solving.

According to the Cloud perspective, composing is as follows:

1. Airlines have fixed contracts with ground handling companies (SP). Each flight is regarded as a service request to the SP.
2. This request can only be fulfilled by a composite service; this service includes all subservices required for handling aircrafts.
3. The SP can use internal resources for delivering the subservices; each resource is being represented as a service. In addition, the SP can ask a third-party SP for external resources (outsourcing), thus for replacing or supplementing its internal resources, e.g., to cope with bottlenecks. In the former case, the SP would establish an SLA with its internal resource; in the latter case, with the third-party SP.

In ground handling, the price for a service depends on expected consumption of resources and also required QoS level. The most important QoS parameter is execution time; for instance, turn-around time of aircraft: 30, 35, 40 minutes etc. The "fast" services are potential bottleneck services, because they can be used better in peak-time situations to fulfill a critical task. However, accelerating such services by adding resources is hardly possible, unlike "slow" services which provide an implicit potential for acceleration (e.g., by increasing workflow of staff). In our simulation, we assume a fixed execution time per resource type for all service providers, who determine their bid price based on (marginally) different internal linear costs.

Table 2 summarizes the Cloud perspective on the ground handling supply chain system.

TABLE 2
CLOUD PERSPECTIVE ON GROUND HANDLING

Element	Ground Handling Supply Chain System
Resource type	Physical and human resources owned by ground handling companies.
Service type	Airlines consume ground handling as a composite service.
Number and specificity of resources	High, fixed number of resources; low specificity due to standardized service requirements (e.g., by IATA regulation).
Reconfiguration	Adding/removing human resources limited by shift plan; physical resources are bound to airport site.
Pay-per-use model	Airlines pay ground handling services based on mixed price models including consumption and QoS level.
Infrastructure Provider (IP)	Represents upstream tier of a ground handling company, i.e., its resources.
Service Provider (SP)	Represents ground handling company providing services to airlines being the final customers.
Service Level Agreements	Exist for contracts between (1) airline and ground handling company, (2) ground handling company and resource, and (3) ground handling company and third-party ground handling company.

B. Instantiation of Model Elements

Next, we instantiate the proposed model elements for the ground handling supply chain. Table 3 lists the main model elements. We limit the scope to a small snapshot of the entire ground handling process, which often includes more than 30 different service types and a respectively high number of relationships.

TABLE 3
INSTANTIATION OF MODEL ELEMENTS

Model Element	Instance Data
Actors (A)	Resources (IP), owned by ground handling companies (SP), airlines.
Service flows (S)	SP to airline (per flight); IP to SP (per service type of resource); SP to SP (per outsourcing job).
Composite services (CS)	For each flight, with WF dependent on direction (inbound/outbound).
Service classification (C)	Domain-specific classification, due to limitations of the SCOR model.
Service types (P)	Five elementary types, based on three resource types (bus: passenger transportation service; baggage cart: delivery to aircraft, pick-up at aircraft; loading staff: loading aircraft, unloading aircraft).
Service data (D)	Data elements: flight number, start time, end time, aircraft type, number of passengers, position, terminal.

When adopting the SCOR model for classifying ground handling services, it appears that all services are of the same type: On the first SCOR level, they are delivery services (D); on the second level, they are of make-to-order (D2), because their intangibility prevents storage. Therefore, we apply a more specific domain classification, based on IATA [23]. We define the most generic type p_0 and subtypes as shown in Fig. 7; the index refers to the service category of IATA.

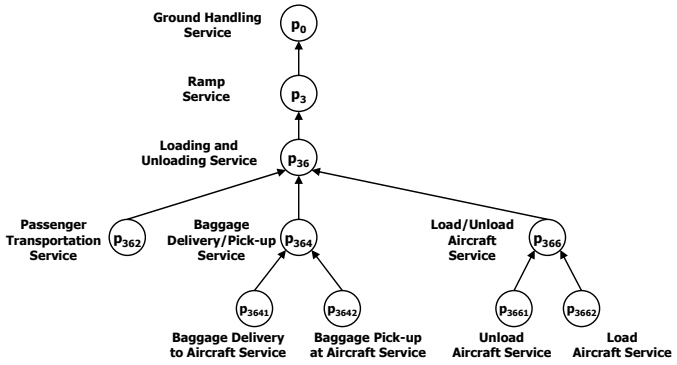

Fig. 7. Classification of Ground Handling Services.

Composite services need to be distinguished for inbound and outbound flights. For an inbound flight, the composite service is $cs_{in}=(a_1, a_2, CE, WF)$ with:

1. a_1 being a SP and a_2 being an airline,
2. *CE* containing elements of four service types, thus $\varphi ce_1 \Im CE{:}p(ce_1)=\{p_{362}\} \omega \varphi ce_2 \Im CE{:}p(ce_2)=\{p_{3641}\} \omega \varphi ce_3 \Im CE{:}p(ce_3)=\{p_{3661}\} \omega \varphi ce_4 \Im CE{:}p(ce_4)=\{p_{3642}\}$
3. $WF=(A', S', a_1)$ with all services s' consumed by a_1 and no service delivered by a_1, and the following sequence: $<p_{3641}, p_{3661}, p_{3642}>$ (i.e., deliver empty baggage carts to aircraft, unload aircraft, pick-up and transfer to terminal) parallel to p_{362} (passenger transportation).

For an outbound flight, the composite service is similar, with *CE* containing only three service types, and workflow sequence $<p_{3641}, p_{3661}>$ (i.e., deliver baggage carts to aircraft, load aircraft) parallel to p_{362}.

Service data extends the service classification and description in ground handling. Some core data elements exist, which describe any type of ground handling service; these data elements are used by all actors for describing services. In the following, we consider as data elements: flight number (used as an identifier per day), start time (determining

earliest start time of the service), end time (determining latest end time of the service), aircraft type (determining equipment and handling procedure), number of passengers (determining number of resources required), position (determining place of service delivery), and terminal (determining source and/or destination).

All these elements describe requirements on the composite service requested by an airline. We can use the concept of data types *(D)* forming a type hierarchy introduced before to determine allowed matches between a service request and service offers. For instance, a service request for handling an Airbus A340-400 can be fulfilled by any service offering which either exactly denotes this aircraft type or one of which 'A340-400' is a subtype, i.e., 'A340'. The same is true for data element 'position' (e.g., subdivided into remote and nose-in) and 'terminal' (subdivided into geographical sites).

A *Semantic Link* defines the data flow within a composite ground handling service. Let's first study the workflow of outbound flights: There are two parallel flows for passengers and baggage. These flows are independent from each other, with the only precondition being the aircraft ready for handling, indicated by the earliest start time (which may differ for the two flows); hence, no data flow necessary. The sequence $<p_{3641}, p_{3661}>$ requires a data flow, because the delivery service hands over the number of baggage items to the load service. Due to the sequence, another flow can be identified: The actual end time of p_{3641} (output) re-determines the earliest start time of p_{3661} (input) (assuming no overlap of the two services).

With regard to inbound flights, the sequence $<p_{3641}, p_{3661}, p_{3642}>$ requires transferring the number of baggage items from the unloading service to the pick-up service, with two instances of the 'actual end time' to 'earliest start time' hand over.

When the composite service has been delivered, it returns output data. This output data describes the service delivery; here, it returns the same input data elements with their actual values, e.g., actual start time, actual end time, and actual number of passengers. Depending on the data type and workflow, different data transformations take place. For instance, the number of passengers is the sum of all passengers transported, whereas the end time is the maximum of the latest end time of all contained service, and other data elements are not being affected at all (e.g., flight number). Such transformations can be categorized by adopting the well-known workflow patterns of van der Aalst et al. [24].

A *SIDI* enables actual data flows, if the output data does not correspond with the required input data, or is missing at all. In ground handling, such a situation is normally excluded by strict ground handling process definitions (for instance, often given by airlines and involving auditing processes before implementation). However, in cases of ad-hoc outsourcing of services to third parties, this assumption is not longer valid, and mismatches of data can happen. Therefore, a SIDI can support end-users in correcting or filling in data by means of user interfaces on mobile devices. Since the number of potential data elements is low and all elements concern service characteristics which can be observed by the service staff and are part of their professional work, the requirements on end-user support are also rather low.

C. Prototype Implementation

We use the proposed models for solving a specific coordination problem in the supply chain system introduced before. For this purpose, we implemented a software prototype based on the technology of SOC and Cloud Computing. Next, we report briefly about the implementation and results.

First, we define the coordination problem: The ultimate goal of the supply chain system is to effectively handle all flights (i.e., SP fulfills all SLAs with airlines) while efficiently using the available resources (i.e., SP minimizes incurred costs).

In general, a SP can implement a variety of approaches for determining the optimal resources to be used, e.g., by planning algorithms. However, SOC aims at loosely coupled services with no central entity for planning service delivery in a hierarchical manner. The service-oriented approach transforms passive resources into active services, which also follow and implement certain strategies (e.g., uniform resource usage per day, offsets between proceedings jobs). Therefore, we need a coordination mechanism that pays attention to potentially conflicting strategies of both SPs and IPs. A candidate group of coordination mechanisms is market-based coordination (i.e., exchanges, auctions) which uses the price for matching demand and supply. Due to the 1-to-many relationship of demand and supply, we choose *reverse auctions* for allocating resources.

The individual strategies and bidding behavior of supply chain actors are represented by cooperative software agents; thus software agents negotiate about the delivery of supply chain services. Cooperative software agents and multiagent systems are suitable technology for solving coordination problems in and between enterprises [25]. These negotiations take place in a service-oriented environment, in particular, within the SOC technology stack for Web services and using a Grid middleware. Table 4 gives an overview of the service-oriented technologies and specifications used.

TABLE 4
SERVICE-ORIENTED TECHNOLOGIES USED

Area	Usage in Prototype
Service Description	WSDL is the language for technically describing supply chain services as Web services.
Service Discovery	All services registered in a Web service registry, which can be queried by each SP for services.
Service Matching	Matching between request and offer is delegated to the reverse auction protocol.
Service Contracts	SA-SLA is the language for semantically describing SLAs in the supply chain system.
Service Execution	Execution of supply chain services is simulated by given execution time per service.
Service Communication	SOAP is the protocol for all communication between SPs and IPs.
Service Middleware	Globus Toolkit 4 [26] is used for developing and executing the software-based services.

The prototype system was used for conducting several experiments: Real-world operational data from an airport company was retrieved and loaded into the system; it included flight plan data, ground handling workflow definition, and a set of workflow rules (e.g., related to aircraft types, airlines, etc.). We then setup an experimental ground handling supply chain, consisting of three SPs, 58 resources of three different types, providing the five service types, and 15 airlines. The flight plan represented a so called peak-time with expected shortage of resources causing bottlenecks and thus the need for adapting the pre-planned service delivery (82 flight movements over a period of three and a half hours).

The coordination mechanism is as follows: Starting with a comprehensive initial plan for the whole day (retrieved from an airport planning system), events triggered by aircraft movements occur; for instance, notification about an incoming flight. The system checks each event for potential effect on the plan. In case of a conflict, e.g., delay of service execution not possible, because the resource is already assigned to another flight, the three-step allocation process starts. First, internal resources are tried by starting a reverse auction. If this fails, the system outsources the task to a third party by an interorganizational reverse auction. The auctions are combinatorial, thus may include sets of interrelated services for the same flight. Service providers can then submit bids on single or combined services. The sequence of services in the right order and at the right time is given in an abstract workflow which is public to all actors.

Due to the market-based approach, the system does not give priorities to flights (customers) in advance, but delegates this function to SPs and their IPs. Since each SP aims at fulfilling its SLAs with airlines, the bidding strategy of their IPs is maximizing revenue and avoiding penalty, both as defined in airline SLAs and propagated to the upstream providers.

The experience made was that the service-based system successfully can handle deviations, as they occur regularly in ground handling, and that the coordination mechanism provides a mean for (1) reallocating internal resources, (2) outsourcing jobs to third party SPs, and (3) considering SLAs that exist with the final customer (airlines). A detailed description of the prototype system, the experiment, and its results can be found in [27]. While the results must be largely attributed to the coordination mechanism, which is out of scope of the present work, it gives an indication of contributions that can be expected from adopting the service-oriented paradigm and its technologies: We found out that real-world supply chain services can effectively be transformed into Web services. If this step is made, then the infrastructure of SOC and Cloud gets available for supply chain systems as well. In the prototype system, we reused and extended a service middleware which is originally and foremost designed for coordinating service delivery in high performance computing as the traditional domain of Grids.

The implementation and experiments show several potential outcomes of transforming supply chain operations into Web services: one is increased accessibility of supply chain services via standardized electronic interfaces. In our case, a uniform access to available resources of third parties, in particular of small and medium-sized companies, at airports is hardly possible and requires relatively high integration efforts. This benefit was identified by end-users. Closely related is the principle of pooling shared resources; by such, the solution space for solving short-term allocation problems can be increased. With regard to performance measurement, the results indicate a slightly better usage of some resources, thus reducing idle times. However, other resources are used less; more experiments are required to test the validity of this effect. Another outcome is due to outsourcing of services: the system handles bottlenecks effectively by generating a valid plan in any tested case, whereas the conventional planning system stops and requires manual plan repair by qualified staff. The contribution from Cloud Computing to this observation is the technical infrastructure allowing such outsourcing.

VI. Related Work

Supply chain systems have been subject to much research in Computer Science, Information Systems, and most recently Service Science. Of particular interest are works on adopting ideas, concepts, models, and methods from software technology which follows the service paradigm, i.e., Grid Computing, Service-oriented Computing, and

Cloud Computing. One can identify two major research streams: The first stream is adopting the service paradigm for supply chain management software. This approach aims at providing the functionality of SCM software by an integrated set of (web) services, e.g., [28]. The second stream is adopting the service paradigm for representing supply chain systems – or parts of it – by (web) services. In this sense, the electronic representations become the subject of Web service coordination and its respective body of knowledge. The work presented belongs to the latter stream.

Literature supports that software-based services contribute to a supply chain system's performance. Kumar et al. study SOC adoption by 388 US firms [29], without distinguishing service types. The ability to dynamically outsourcing tasks as Web services is emphasized by Iyer et al. which explore the strategic implications on supply chain systems being transformed into dynamic business networks [30].

Blau et al. propose the concept of Service Value Network (SVN) [31]; it represents a network of business entities that provide business value through market-based composition of complex services from a pool of standardized service modules. The similarity to our approach is the conceptualization of service and composition; the formalism used is based on statecharts. The objective of this research is different from ours and hence its perspective of market mechanism design, i.e., game and auction theory. In addition, it concerns service industry, but not supply chain systems.

Gujo proposes a combinatorial multi-attribute auction model for allocating logistics services in a network of logistics service providers [32]. Transportation jobs are conceptualized as service requests, for which providers can submit combinatorial bids representing service bundles, i.e., for each tour. While the coordination mechanism is relevant to our work, the services are independent from a computing infrastructure and not motivated by service-orientation. Another market mechanism is used by Bossenbroeck et al. [33], who propose option contracts to reducing uncertainty of resource availability and volatility of resource prices; though this approach is confined to computational resources.

In recent years, there have been efforts within Grid Computing to extend its coverage to business applications and also supply chain systems. For instance, the European initiative BEinGRID conducted several business experiments (BE) [34]: BE17 concerns optimization in retail supply chains and thus provides supply chain functionality as services; BE22 concerns food supply chains and adopts the Grid's concept of Virtual Organization (VO) for representing a short-term configuration of supply chain participants [35]. This approach is situated in the Grid technology software stack and adopts SLAs for supply chain contracts. Details about its service representation and potential compositions have not been published though.

SLAs are of high importance in service-based supply chains. Müller et al. study the expressivity of SLA languages with regard to temporal dependencies existing between logistics services and extend the WS-Agreement specification to allow for validity periods of SLA terms [36]. In prior work, we proposed a SLA-based negotiation protocol for service supply chains [37]; the objective is to prevent over-commitments and over-purchasing by the Service Provider. The present work adopts its basic supply chain service definition and extends it with regard to composition, classification, and data flows.

To the best of our knowledge, supply chain systems have not yet been analyzed from the perspective of Cloud Computing. Following the first research stream, Cloud is bound to providing computational services of supply chain software functionality such as planning, but not operations.

VII. Conclusions

Coordinating resources and processes is a severe problem in supply chain systems. In this paper, we proposed 'Supply Chain as a Service' as a service-oriented approach to its solution. The rationale is as follows: Cloud Computing is in the process of providing a viable, universal middleware for delivering services to end users. Making this infrastructure accessible to supply chain systems by means of 'servitizing' is beneficial to supply chain coordination, because of loosely coupled services and respective middleware standards.

We adopted the idea of Cloud Computing for service representation and service composition in supply chain systems. The resulting models allow for representing operations in supply chain systems as interconnected services and describing compositions of such services. A first evaluation in a supply chain of airport ground handling operations shows evidence that the proposal provides a core set of constructs for describing supply chain systems and – on top that – coordinating service delivery to the final customer.

The proposal has limitations. First, by adopting the basic idea of Cloud Computing, the proposal also inherits some of Cloud's deficits. For instance, Cloud Computing is still in an early stage, with few matured concepts and also remaining confusion about its paradigm and contributions. Second, the proposal is focused on service description, their interrelations and interoperability, but does not explore the problem of composing services, thus finding optimal compositions; though it provides a model basis for the creation of service composition approaches. Third, the evaluation is limited to a specific use case and thus lacks evidence of general suitability and validity of the proposed

models. Fourth, the proposal is aimed at the conceptual level of service description, but not integrated into the SOC technology stack of specification languages, such as WSDL, OWL-S, WS-Agreement, SA-SLA, etc.

Therefore, future work is aimed at extending the coverage of the proposed Cloud adoption; the current work is regarded as providing a framework for it. In particular, research is needed on (1) adopting formal languages from SOC for specifying SLAs in supply chain systems, (2) selecting and adopting service composition algorithms suitable for supply chain services, (3) integrating supply chain service description with models originating from Semantic Web Services, and (4) testing the approach in more realistic and comprehensive scenarios of supply chains and providing respective quantitative evaluation.

ACKNOWLEDGMENT

The first two authors would like to thank Paul Karaenke and Michael Schuele for their assistance in designing and implementing the prototype system. We thank Andreas Scheuermann for his helpful comments on an earlier version of this manuscript.

REFERENCES

[1] A.M. Sánchez and M.P. Pérez, "Supply chain flexibility and firm performance: A conceptual model and empirical study in the automotive industry," *Int. J. Operations & Prod. Manage.*, vol. 25, no. 7, pp. 681−700, 2005.
[2] B. Avittathur and P.M. Swamidass, "Matching plant flexibility and supplier flexibility: Lessons from small suppliers of U.S. manufacturing plants in India," *J. Operations Manage.*, vol. 25, no. 3, pp. 717−735, 2007.
[3] B. Hayes, "Cloud computing," *Commun. ACM*, vol. 51, no. 7, pp. 9−11, 2008.
[4] L.M. Vaquero, L. Rodero-Merion, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Comput. Commun. Review*, vol. 39, no. 1, pp. 50−55, 2008.
[5] G. Stevens, "Integrating the supply chain," *Int. J. Physical Distribution & Materials Manage.*, vol. 19, no. 1, pp. 3−8, 1989.
[6] J.T. Mentzer, W. DeWitt, J.S. Keebler, S. Min, N.W. Nix, C.D. Smith, and Z.G. Zacharia, "Defining supply chain management," *J. Bus. Logistics*, vol. 22, no. 2, pp. 1−16, 2001.
[7] L. Garicano and S.N. Kaplan, "The effects of business-to-business E-commerce on transaction costs," *J. Industrial Economics*, vol. 49, no. 4, pp. 463−485, 2001.
[8] M. Rappa, "The utility business model and the future of computing services,", *IBM Syst. J.*, vol. 43, no. 1, pp. 32−42, Jan. 2004.
[9] M. Reimann, O. Schilkeb, and J.S. Thomas, "Toward an understanding of industry commoditization: Its nature and role in evolving marketing competition," *Int. J. Research in Marketing*, vol. 27, no. 2, pp. 188−197, 2010.
[10] T.H. Davenport, "The coming commoditization of processes," *Harvard Bus. Review*, vol. 83, no. 6, pp. 100−108, Jun. 2005.
[11] P. Matthyssens and K. Vandenbempt, "Moving from basic offerings to value-added solutions: Strategies, barriers and alignment," *Ind. Marketing Manage.*, vol. 37, no. 3, pp. 316−328, 2008.
[12] D. Verma, *Supporting Service Level Agreements on IP Networks*, Arenda, Norway: Macmillan Technical Publishing, 1999.
[13] Open Grid Forum, *Web Services Agreement Specification*, March 2007.
[14] A. Cox, "The art of the possible: relationship management in power regimes and supply chains," *Supply Chain Manage.: An Int. J.*, vol. 9, no. 5, pp. 346−356, 2004.
[15] M. Jaeger, G. Rojec-Goldmann, and G. Mühl, "QoS Aggregation for Web Service Composition using Workflow Patterns," in *Proc. 8th IEEE Int. Enterprise Distributed Object Computing Conf.*, Monterey, CA, 2004, pp. 149-159.
[16] World Wide Web Consortium (2010). *Web of Services* [Online]. Available: http://www.w3.org/standards/webofservices
[17] F. Casati and M.-C. Shan, "Dynamic and adaptive composition of e-services," *Inform. Syst.*, vol. 26, pp. 143–163, 2001.
[18] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *IEEE Computer*, vol. 40, no. 11, pp. 38-45, Nov. 2007.
[19] W3C (2010). *OWL-S: Semantic Markup for Web Services* [Online]. Available: http://www.w3.org/Submission/OWL-S.
[20] M. Paolucci, T. Kawamura, T.R. Payne, K. Sycara, "Semantic Matching of Web Services Capabilities," in *Proc. Int. Semantic Web Conf. (ISWC 2002)*, Sardinia, Italy, 2002.
[21] Supply-Chain Council (2010), *Supply Chain Operations Reference Model (SCOR®), Version 10.0* [Online]. Available: http://www.supply-chain.org
[22] T. Schlegel, W. Beinhauer, and F. Meo, F., "Object-orientation in planning and control of decentralized production systems," *VIMation Journ. Human-Machine Interaction and Models in Prod.*, no. 1, pp. 54–63, 2008.
[23] International Air Transport Association, *IATA Standard Ground Handling Agreement*, Montreal, Canada, 2004.
[24] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 3, 2003, pp. 5–51.
[25] S. Kirn, O. Herzog, P. Lockemann, and O. Spaniol, Eds., *Multiagent Engineering – Theory and Applications in Enterprises*. Berlin:Springer, 2006.
[26] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems," in *IFIP Int. Conf. Network and Parallel Computing*, Springer LNCS 3779, 2005, pp 2–13.
[27] I. Jones, Ed., *BREIN Final Demonstrator*, Deliverable D8.1.1 [Online]. Available: http://www.eu-brein.com.
[28] J.C.P. Cheng, K.H. Law, A. Jones, and R. Sriram, "Service oriented and orchestration framework for supply chain integration," in *Proc. ASME 2009 Int. Design Eng. Tech. Conf. & Computers and Inform. Eng. Conf.*, San Diego, CA, 2009.
[29] S. Kumar, V. Dakshinamoorthy, and M.S. Krishnan, "Does SOA improve the supply chain? An empirical analysis of the impact of SOA adoption on electronic supply chain performance," in *Proc. 40th Annu. Hawaii Int. Conf. Syst. Sciences*, Big Island, HI, 2007.
[30] B. Iyer, J. Freedman, M. Gaynor, and G. Wyner, "Web services: enabling dynamic business networks," *Commun. Assoc. Inf. Systems*, vol. 11, no. 1, 2003.
[31] B. Blau, C. van Dinther, T. Conte, Y. Xu, and C. Weinhardt, "How to coordinate value generation in service networks – a mechanism design approach," *Bus. and Inform. Systems Eng.*, vol. 5, no. 1, pp. 343–356, 2009.
[32] O. Gujo, "Multi-attribute inter-enterprise exchange of logistics services," in *Proc. 10th IEEE Int. Conf. E-Commerce Technology (CEC 2008)*, Washington, D.C., 2008, pp. 113−120.

[33] A. Bossenbroek, A. Tirado-Ramos, P.M.A. Sloot, "Grid Resource Allocation by Means of Option Contracts," *IEEE Sys. J.*, vol. 3, no. 1, 2009, pp. 49−64.
[34] BEinGRID (2010). *Business Experiments in Grid* [Online]. Available: http://www.beingrid.eu.
[35] E. Volk, M. Mueller, A. Jacob, P. Racz, and M. Waldburger, "Increasing capacity exploitation in food supply chains using grid concepts," in *Proc. Grid Economics and Bus. Models Workshop*, Delft, 2009, pp. 88−101.
[36] C. Müller, M. Resinas, and A. Ruiz-Cortés, "Using automated analysis of temporal-aware SLAs in logistics," in *Proc. 1st Int. Workshop on Service-Oriented Computing in Logistics*, Stockholm, 2009, pp. 156-164.
[37] P. Karaenke and S. Kirn, "A multi-tier negotiation protocol for logistics service chains," in *Proc. 18th European Conf. Inform. Syst.*, Pretoria, 2010, pp. 1559-1560.