A Service-Oriented Approach to Freight Routing in Intermodal Transport Systems

Joerg Leukel
University of Hohenheim
Schwerzstrasse 35
70599 Stuttgart, Germany
+49 711 459-23968
joerg.leukel@uni-hohenheim.de

Stefan Kirn
University of Hohenheim
Schwerzstrasse 35
70599 Stuttgart, Germany
+49 711 459-24025
stefan.kirn@uni-hohenheim.de

ABSTRACT

Determining optimal routes for given freight is a core decision in logistics. In intermodal logistics, freight routing has to consider the interfaces between different modes of transportation, such as hand-over offsets, load changes, and organizational procedures. We study this problem from the perspective of Service-Oriented Computing (SOC). We (1) propose representing intermodal transport systems as a set of service offerings and customer demand as service requests, (2) define freight routing as a service composition problem, and (3) develop a composition algorithm for transportation services.

Keywords

Logistics, Service Composition, Service-Oriented Computing.

1. INTRODUCTION

Freight transport systems are challenged by increasing requirements from supply chains and markets. These requirements concern their throughput, scalability, and flexibility to meet growing and individual customer demand. Intermodal transport systems are of particular importance, since they serve as the backbone of global trade [26].

The vital role of information technology for coordinating resources and activities in transport systems has been acknowledged a long time ago [9]. On the operational level, freight routing, thus the process of selecting the best route for given freight, can be supported by acquiring information about existing relations, assessing the transport system formed by these relations, and determining routes by linking and instantiating relations [3][7]. Such support, however, depends on the availability, accurateness, and interpretation of such information.

Intermodality causes often additional costs and delays at the interface between modes. Overcoming *organizational and technical barriers* of intermodality can be achieved by explicitly describing intermodal exchanges and integrating these descriptions into decision making. This paper addresses these barriers by grounding freight routing on key concepts and formalisms of Service-Oriented Computing (SOC). As a paradigm for software systems, SOC aims at rapidly and easily developing applications by composing single services. A service is an autonomous, platform-independent computational entity that provides some functionality via an interface [16].

The current state of SOC adoption in logistics is that of a paradigm which transforms existing software architectures into service-based systems (e.g., [11]). IS research has attributed these architecture with better supporting flexibility of business processes [6][10]. Unlike the dominating computational SOC approach, which regards electronic services as means of logistics IT functionality, such as resource planning, we represent *transport operations* as software-based services. Thus we do not represent logistics planning functionality, but use the SOC idea for a new class of software-based services. These services match directly to services in economics and give access to operations within a logistics system by means of standardized electronic interfaces. This understanding of services is a constituent of an overarching research program that studies coordination problems in multi-tier supply chains for fulfilling individual demand. In the research at hand, individual demand is that of a routing request for which a solution is not given a-priori, but must be determined based on available service offerings.

In particular, we (1) propose representing intermodal transport systems as a set of service offerings and customer demand as service requests, (2) define freight routing as a service composition problem, and (3) develop a composition algorithm customized for transportation services. We demonstrate the applicability and usefulness in a simulation experiment. The contribution is service-oriented freight routing which takes into account barriers between relations.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 provides a basic model of intermodal transport. Section 4 introduces the SOC perspective and develops service-oriented routing. Section 5 provides an evaluation and discussion. Section 6 provides a short conclusion.

10th International Conference on Wirtschaftsinformatik,
16th - 18th February 2011, Zurich, Switzerland

2. RELATED WORK

The relevant existing body of knowledge originates from two major areas: intermodal logistics and service composition.

2.1 Intermodal Logistics

Intermodal logistics has been given a top priority in many regions due to its significant growth [8]. Routing is a classic task in logistics and other network systems, e.g., telecommunication systems. Intermodal freight routing is different from vehicle routing, because it incorporates mode changes and often uses relations by more than one logistics service provider (LSP). Referring to the classification by *Caris et al*. [5], the time horizon of this decision problem is operational and the decision maker is the intermodal operator, e.g., shipper.

In parallel to the growing economic importance of intermodal logistics, interest in routing has increased. The early work by *Min* [14] uses goal programming for solving the conflict of costs, delivery time, and service quality between alternative modes. This approach is, however, severely limited: transport time is proportional to the distance traveled by each mode, transport units are fixed to one container size for all modes, and modal transfers are always intra-organizational. These limitations, in particular the latter, exclude competitive intermodal transport systems.

Boardman et al. [3] apply a k-shortest path algorithm. This approach considers transport costs per relation, transfer costs between modes, and delivery time, whereas it abstracts from concrete offerings of LSPs. Therefore, for each relation and mode exists only one pre-defined LSP; hence it is not possible to select from competitive LSPs. In [4], optimality is defined as solving a two-objective problem (minimize time, minimize costs). The shortest path algorithm is applied to a transport systems consisting of five major Canadian, three Mexican destinations, and several modes for each relation. This research determines concrete optimal routes without providing a general approach to selecting offerings by LSPs. The work of *Chang* [7] considers time windows of transport modes. It also assumes more realistic cost functions (concave instead of linear). The proposed heuristic aims at minimizing time and cost, but restricts the transport unit to one type only (air freight container). *Ziliaskopoulos* and *Wardell* [25] include an even larger set of constraints: delays at modal changes and time-dependent travel times. Their algorithm indicates an almost linear complexity for passenger transports, but it is tested for systems with three predecessors per node only.

2.2 Service Composition

Determining service compositions still remains an important issue in SOC research [16]. With regard to our approach, two major approaches from SOC are relevant: (1) semantic service description, and (2) template-based composition.

Semantic service description enriches service descriptions in such a way that one can determine compositions by reasoning about pre- and post-conditions and other service parameters of available services. It requires the description of all services based on a common service ontology. Description languages such as OWL-S have been proposed. A key requirement for service composition is the consideration of the quality of service (QoS). Unlike approaches such as BPEL, Petri-net, and pi Calculus, these parameters are part of all semantic approaches [13]. By following the semantic direction, our proposal considers logistics QoS.

A comprehensive QoS-aware composition framework is proposed by *Zeng et al.* [24]. Users define business objectives that must be reached. Then the system generates the required services for these objectives based on a set of domain-specific business rules; additional rules are applied to create chains of services, by linking, adding, and removing services subsequently. The difference to our approach is that it allows specifying user requests very powerful, e.g., by providing parts of an abstract workflow. However, it requires the codification of all domain knowledge by three types of rules.

A sub-task of composition is checking the validity of two linked services (sequence). Matching types for describing the match between output of the preceding service and input of the proceeding service were proposed by *Lecue* and *Delteil* [12]. We will use this linkage for transport services.

Template-based composition relies on domain-specific templates, which are abstract workflows. A composition algorithm then instantiates and/or modifies the template. This avenue of research is similar to ‘pattern-based workflow generation’ [23], which also determines workflows by reusing knowledge about the domain. The most general knowledge can be retrieved from *van der Aalst et al.*’s [1] workflow patterns, which are fully domain-independent. Applying this idea to service composition can also be denoted as configuration; this term emphasizes that the search space is reduced. *ten Teije et al.* [19] exploit specific knowledge about objects and propose an algorithm for filling a template. The major differences of our work are that (1) we allow modifications of the template by inserting parallels and loops and (2) provide a richer semantic service model.

Service composition has also been acknowledged by IS research: *Blau et al.* [2] propose the concept of Service Value Network; 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. *Röglinger* [18] proposes an operationalization of correctness for service compositions and thus contributes to formally measuring and ranking alternative compositions; this research does not integrate itself into semantic service models.

3. BASIC MODEL

This section defines a basic model of intermodal transport system and transport service. It will be used and extended in the subsequent section. We also define assumptions of our work.

3.1 Intermodal Transport System

A transport system is a logistics system concerned with delivering goods from origins to destinations. It consists of nodes participating in transforming goods with regard to location, time, and quantity. Nodes are inter-connected by relations (possible flow of goods), whereas nodes represent the transshipment of freight (e.g., terminal). An intermodal transport system consists of at least two different modes of transport (e.g., road and rail transport), which requires transshipment while most often keeping the transport unit (e.g., by standard containers).

(Definition 1) Intermodal transport system is a directed graph $ITS=(N,R,C,D)$, where N is the set of nodes and R is the set of relations with $R \subseteq N \times N \times TU \times M$. Each $r \in R$ is a 4-tuple $r=(n_i,n_j,tu,m)$, with flow of transport unit $tu \in TU$ from $n_i \in N$ to $n_j \in N$ using the mode $m \in M$. C is a function that defines the cost $c_{x,y}(n)$ for a transshipment from mode $x \in M$ to mode $y \in M$ at node $n \in N$, i.e.:

$$C \in x, y, n \rightarrow \mathbb{R}_{>0}$$

Respectively, D is a function that defines the delay $d_{x,y}(n)$ for transshipment from mode $x \in M$ to mode $y \in M$ at node $n \in N$, i.e.,

$$D \in x, y, n \rightarrow \mathbb{R}_{>0}$$

3.2 Transport Service

Transport services are offered by LSPs and consumed by, e.g, shippers. A transport service realizes at least one relation $r \in R$ in *ITS*. As such, it can be regarded as an abstraction from the underlying physical infrastructure. The set of all logistics services is captured in the transport service flow model.

(Definition 2) Transport service flow model is a directed graph $TSF=(A,S,MA)$. A is the set of actors. S is the set of offered transport services. Each $s \in S$ is a tuple $s=(a_j,a_k)$, with flow s from $a_j \in A$ to $a_k \in A$. MA is a (mathematical) relation which maps each s to relations R in *ITS*, i.e., $MA \in S \rightarrow R$. Thus each transport service s can implement $|MA(s)|$ relations in *ITS*.

Integrity constraints must hold for existence of actor who does not provide a service (customer only), actor who does not consume a service (LSP only), and weak connectivity. In addition, it has to be assured that a service can only be mapped to more than one relation, if all such relations are connected by a walk. Therefore: For any s, if $|MA(s)| \geq 2$, then $t:=|MA(s)|$ and there must exist a walk w_s in *ITS* with $w_s=(n_i, ma_{s,1},.., ma_{s,t}, n_j)$ and $n_i,n_j \in N$.

3.3 Optimal Freight Route

Freight routing is the process of selecting the most appropriate route for shipments through the transport system. The optimal route is the one which best fulfills the request. Ultimately, optimality of a route can be reduced to minimizing its costs. If more criteria, such as time and alpha service level, are used, then these criteria could be weighted and their values aggregated into total weighted costs.

(Definition 3) Routing request is defined as $req=(n_{origin}, n_{dest}, tu)$ for transport of $tu \in TU$ from $n_{origin} \in N$ to $n_{dest} \in N$.

(Definition 4) Optimal freight route is a way w^* in *ITS* with $w^*=(n_{origin}, r_1,.., r_k, n_{dest})$, and minimizing its costs $c(w^*)$:

$$c(w^*) = min \sum_{i=1}^{k} c(r_i)$$

4. SERVICE-ORIENTED ROUTING

This section proposes solving the freight routing problem by composing transport services. We introduce and adopt SOC formalisms for describing services and their composition.

4.1 Rationale

SOC defines a service as an “autonomous, platform-independent computational entity” providing some functionality, which can be accessed over an infrastructure via an interface [16]. Service usage takes place by exchanging messages between service provider and service customer. In technical terms such services are Web Services (WS) being implemented on the WS technology stack. We adopt this definition to transport service. The aim is to *represent* transport services – which are executed physically in logistics – as electronic services. Calling the electronic service means submitting an order to the LSP. Response messages by the provider inform the customer about the status of service execution; e.g., delivery advice, delivery notification etc.

In a naïve scenario assume that each relation $r \in R$ in *ITS* is represented by exactly one electronic service. Then a route from node n_1 via n_2 to n_3 can be defined by (1) selecting the service, which connects n_1 and n_2, and the service, which connects n_2 and n_3, and (2) linking these services in sequence. This combination of services is called *composition* or *composite service*. The latter term emphasizes that the composition itself is a service that can be offered by a service provider. The logical structure of a composition needs to be described in a *workflow*. The workflow can either be given by the service customer (in case the composition is known) or must be determined by the service provider. The latter case matches to the freight routing problem.

What makes determining the composition difficult is the complexity and diversity of service offerings. Intermodal logistics is characterized by at least two modes of transport, thus regularly two specific services need to be combined. Complexity refers to the number of service offerings and number of dependencies between offerings. For instance, the physical infrastructure used must be considered when composing services (e.g., transferring transport units, loading/unloading of vehicles). Organizational procedures play also an important role (e.g., time slots for delivery, qualification of staff).

These conditions constrain the set of valid compositions. This class of composition is addressed in SOC by a semantically rich description of services. A common model is IOPE (inputs, outputs, preconditions, and effects) as part OWL-S [21], which is an ontology for describing Web services. Respective services are then called Semantic Web Services (SWS).

4.2 IOPE Service Model

IOPE structures the service description into inputs, outputs, preconditions, and effects. The assumption is that all services are described by referring to a domain ontology T. IOPE reflects the service functionality as an information transformation and a state change resulting from the service.

Information transformation is subject of input and output. Valid input can be restricted by referring to a concept of the ontology T. It is important that a transport service represents a physical activity taking place in the real world, thus transformation is not limited to information, but concerns the object of this physical activity; hence the transport unit *tu*. We therefore interpret IO as the *physical transformation*. Further, intermodal transport aims at keeping the transport unit unchanged, e.g, output equal to input.

The *state change* is captured by PE. Preconditions are constraints over inter-dependent input information. In transportation, these need to be related to physical state. At least it is required that the transport unit is located at the service's origin node. We thus formulate an axiom: Let *origin_s* be the origin node of *s*, then *pre_s:=isLocatedAt(tu, origin_s)*. Effect is the change; here it is right the logistics transformation of *tu* in time and location. For the former we define *eff_s:=isLocatedAt(tu,dest_s)*, with *dest_s* the destination node of *s*. The latter is calculated by adding the transport time (*leadtime_s*) to the start of transportation, i.e., *atTime(tu,dest_s):=atTime(tu,origin_s)+leadtime_s*.

Table 1 summarizes the interpretation of IOPE.

Table 1. Interpretation of IOPE

Description	Web Service	Transport Service
Input	Information required for executing the service	Transport unit of the shipper
Output	Information generated by service execution	Transport unit of the shipper
Precondition	Constraint over input information	Disposability of *tu* at origin location
Effect	State change	Transformation of *tu* in time and location

4.3 QoS Model

IOPE supports the finding of *valid* linkages between services. It is, however, not sufficient to determine to which degree the request is fulfilled. This task requires a quantitative assessment by QoS. It amends the service description by non-functional QoS parameters [17]. Next we visit QoS of Web services, and interpret QoS parameters through the lens of transport.

Current specifications for service descriptions do not define QoS parameters a-priori, but delegate this task to service domains. Nevertheless a minimum set of parameters can be identified in SOC literature [17], consisting of execution time, cost, throughput, availability, and reliability. Table 2 maps these parameters to metrics used for measuring transport services.

Table 2. Interpretation of QoS parameters

Parameter	Web Service	Transport Service
Execution time *(et)*	Time between service request and response	Time between pick-up at origin and delivery at destination (lead time)
Cost *(co)*	Cost charged by the service provider	Cost charged by the LSP depending on tariff scheme
Throughput *(tp)*	Number of requests served per time period	Ton kilometers per time period
Availability *(av)*	Time service is available per time period	Time service is available; based on calendar/day.
Reliability *(re)*	Number of correct responses of all responses	Probability that transport arrives at destination in due time (ά service level)

The physical nature of transport has to be considered. For instance, the execution time of a Web service is the time between service request and response. The measurement of transport services is different, because the service starts not at the time of request, but at the time of pick-up at the origin node.

QoS parameters complement the service description. We thus extend the preliminary transport service definition as follows.

(Definition 5) Transport service is a tuple $s=(a_j, a_k, w_{origin}, w_{dest}, m, I, O, P, E, Q)$:

- a_j is the service provider, a_k service consumer, $a_j, a_k \in A$.
- w_{origin} is the start node of walk *w(s)* in *ITS*, w_{dest} is the end node of *w(s)* in *ITS*, with *w(s)* constructed by *MA(s)*.
- *m* is the mode of the last relation in *w(s)*.
- *I* is a set of input information, i.e., transport units $tu \in TU$.
- *O* is a set of output information, i.e., transport units $tu \in TU$.
- *P* is a set of preconditions, i.e., logical axioms.
- *E* is a set of effects, i.e., logical assertions.
- *Q* is a set of QoS parameters with $Q=(Q_{et}, Q_{co}, Q_{tp}, Q_{av}, Q_{re})$.

4.4 Template-based Service Composition

The objective of template-based service composition is to reduce the effort required for finding compositions by incorporating domain knowledge into the search process. Composing is then the process of finding instantiations of each "slot" of a given template [19][23]. We identify templates for intermodal transport systems and formalize them.

4.4.1 Service Composition Templates

Transport systems provide alternative routes for delivering freight from origin to destination. For the task of freight routing the system is given ex-ante. Therefore, we need to examine transport systems for generic structures. The theoretical framework for designing transport systems by *Woxenius* [22] provides six design principles, from which two are relevant for structuring intermodal transport systems:

- *Hub-and-spoke* collects freight in a central node (hub) and then disseminates freight to a number of destinations. Intermodality exists if a mode change takes place at the hub.
- *Connected hubs* extends the hub-and-spoke (Figure 1) by adding relations between two hubs. It subdivides each route into three relations: pre-carriage (to the first hub), main carriage (from first to second hub), and on-carriage (to the destination).

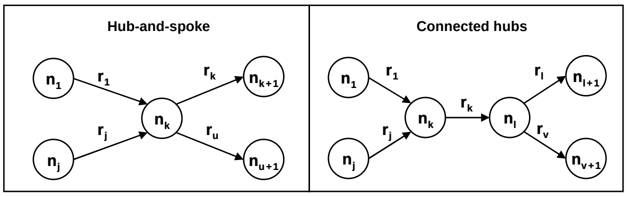


Figure 1. Hub-and-spoke and connected hubs.

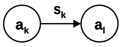
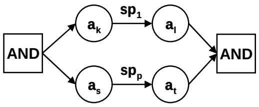
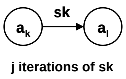
Since routing determines a route from exactly one origin to one destination, we use the number of service slots as a discriminator for composition templates. We start with a template describing a route via a hub, and then enhance the number of service slots.

Let *TH* be a hub-and-spoke composition template $TH=(n_{origin}, n_{dest}, s1, s2)$ connecting $n_{origin} \in N$ and $n_{dest} \in N$ by two subsequent services $s1, s2 \in S$ over a hub $h \in N$.

Let *TC* be a connected hubs composition template $TC=(n_{origin}, n_{dest}, s1, s2, s3)$ connecting $n_{origin} \in N$ and $n_{dest} \in N$ by three subsequent services $s1, s2, s3 \in S$ over two connected hubs $h_1, h_2 \in N$.

These two templates do not cover the full spectrum of possible routes. To further increase the coverage, we study general control structures that can be observed in workflows. *van der Aalst et al.* [1] has exemplified these structures into workflow patterns. Table 3 shows the mapping of relevant patterns to transport services: sequence (#1), multi-merge (#8), and loop (#10). All other workflow patterns are not relevant for *ITS*. For instance, exclusive choice (#4) and simple merge (#5) would make a route non-deterministic.

Table 3. Workflow patterns for transport services

Pattern	Control Flow	Adoption
Sequence (#1)		Freight transport, no split/merge or iteration.
Multi-Merge (#8)	 AND sp_1 a_k a_l sp_p a_s a_t AND	Split of freight at origin into two or more parallel transports followed by merge at destination.
Arbitrary Cycles (Loop, #10)		Iterative transport, if maximum load per service execution lower than load of freight.

We add templates for multi-merge and loop:

Let *TM* be a multi-merge composition template $TM=(n_{origin}, n_{dest}, SP)$ connecting $n_{origin} \in N$ and $n_{dest} \in N$ by parallel services $SP \subseteq S$, with $SP=\{sp_1,..,sp_p\}$ and $p=|SP| \geq 2$.

Let *TL* be a loop composition template $TL=(n_{origin}, n_{dest}, sk)$ connecting $n_{origin} \in N$ and $n_{dest} \in N$ by *j* iterations of $sk \in S$, with $j \geq 2$.

4.4.2 Integrity of Service Compositions

Instantiating a template, thus replacing service slots with actual services, has to guarantee integrity. In transport systems, integrity concerns physical, geographic and time-related conditions of transport units. At least, a timely delivery must be assured. For automating service composition, a formal specification of integrity is needed. This can be achieved by reasoning over semantic service descriptions. We employ the construct of Semantic Links [12] and define integrity constraints for each template.

4.4.2.1 Integrity of Hub Template

The most basic requirement is that two services can be executed in sequence. This structure is found in the hub template as well as in *TC* and *TL*. Semantic Link describes the relationship between output of service *s1* and input of *s2* by function Sim_T:

$$\langle s1, Sim_T(O\_s1, I\_s2), s2 \rangle$$

Sim_T distinguishes five matching types [12]:

- **Exact:** output and input are equivalent, i.e., $T \models O\_s1 \equiv I\_s2$; here: arriving transport unit is equivalent to expected transport unit, thus transport services can be linked.
- **PlugIn:** output is sub-concept of input, i.e., $T \models O\_s1 \sqsubseteq I\_s2$; here: arriving transport unit is specialization of expected transport unit, thus transport services can be linked.
- **Subsume** (output is super-concept of input), **Intersection** (intersection of output and input is not empty), and **Disjoint** (output and input are disjunctive) describe cases where transport services can not be linked.

A sequence of transport services is only possible, if their IO matching type is Exact or PlugIn. Additionally, preconditions and effects must be considered. Effect of *s1* must fulfill precondition of *s2*. Concretely, *s1* must move the transport unit to a destination, which is origin of *s2* (Exact or PlugIn), i.e., $dest\_s1 \sqsubseteq origin\_s2$.

4.4.2.2 Integrity of Connected Hub Template

The *TC* template requires that its three services *s1,s2,s3* can be executed in sequence. Considering the transport unit and location unit, we add constraints over IO and PE as shown in Table 4.

Table 4. Constraints for connected hub template

Description	Constraint
IO	$((O\_s1 \equiv I\_s2) \sqcup (O\_s1 \sqsubseteq I\_s2))$ $\sqcap((O\_s2 \equiv I\_s3) \sqcup (O\_s2 \sqsubseteq I\_s3)($
PE	$(dest\_s1 \sqsubseteq origin\_s2)$ $\sqcap(dest\_s2 \sqsubseteq origin\_s3)$

4.4.2.3 Integrity of Multi-Merge Template

A split may occur in two cases. First, due to load, which exceeds the capacity of a service; in this case, the transport unit of all parallel services is equal. Second, due to lack of a service, which can handle the transport unit solely, thus the freight must be split into other transport units (e.g., container split into several pallets). Since the reason of split is not stated in the template, it does not provide constraints over IO.

Each service requires that the transport unit is located at the same origin; similarly, each service delivers its transport unit to the same destination. These constraints are given in Table 5.

Table 5. Constraints for multi-merge template

Description	Constraint
IO	*None*
PE	$origin\_sp1 \equiv \ldots \equiv origin\_spp$ $dest\_sp1 \equiv \ldots \equiv dest\_spp$

4.4.2.4 Integrity of Loop Template

This template is used if the load exceeds the capacity of a service and thus requires that the same service is executed two or more times. The only dependency between iterations is time-related; there are no constraints over IO and PE.

4.5 Composition Algorithm

We propose an algorithm for template-based service composition. The search space is reduced by (1) domain-specific templates and (2) integrity constraints representing intermodal barriers between relations. First, we determine candidate services for each service slot, built pairs of these candidates, then try to fill slots by a multi-merge or loop structure, if such a structure results in additional valid pairs. The algorithm's input is a request for route as follows.

(Definition 6) Routing request is defined as $req=(n_{origin}, n_{dest}, tu_{in}, tu_{out}, tem, RQ)$ for transport of $tu_{in} \in TU$ from $n_{origin} \in N$ to $n_{dest} \in N$, delivering $tu_{out} \in TU$ by adhering to transport template *tem* with $tem \in \{TH, CTC\}$, and fulfilling QoS parameters *RQ* with $RQ=(RQ_{et}, RQ_{co}, RQ_{tp}, RQ_{av}, RQ_{re})$ (as of Table 2).

4.5.1 Determine Service Candidates

Algorithm 1 collects valid candidates for slot *s1* and *s2*, and returns them in sets *SV1* and *SV2*. For this purpose, all services (line 2) are checked for fulfilling the constraints over IO and PE as well as execution time RQ_{et} and reliability RQ_{re} (line 3/4). It also determines candidates that meet the request, except for throughput; we collect them in sets *SP1* and *SP2* (line 6/11) as potential candidates to be used within a multi-merge or loop.

Determining candidates for slot *s1* has to check origin, transport unit, and QoS parameters. The latter are quantitative parameters. The two former are instances of the underlying domain ontology, thus we can make use of the subsumption relationship. For example, let tu_{in} be a 20-feet container and the service's *tu* an ISO container, then the service fits into the slot, because ISO container is super-concept of 20-feet container ($tu_{in} \sqsubseteq tu$).

Algorithm 1. Service Candidates for Slots *s1* and *s2*

```
1: $SV1:=\emptyset; SP1:=\emptyset; SV2:=\emptyset; SP2:=\emptyset$
2: for all $s \in S$
3:   if ($n_{origin} \sqsubseteq w_{origin}\_s$ and $tu_{in} \sqsubseteq tu\_s$ and $RQ_{av} \sqsubseteq Q_{av}\_s$ and
4:     $RQ_{et} \geq Q_{et}\_s$ and $RQ_{re} \leq Q_{re}$ ) then
5:       if $RQ_{tp} \leq Q_{tp}\_s$ then $SV1:=SV1 \cup s$
6:       else $SP1:=SP1 \cup s$; end if
7:   end if
8:   if ($n_{dest} \sqsubseteq w_{dest}\_s$ and $tu_{out} \sqsubseteq tu\_s$ and $RQ_{av} \sqsubseteq Q_{av}\_s$ and
9:     $RQ_{et} \geq Q_{et}\_s$ and $RQ_{re} \leq Q_{re}\_s$) then
10:      if $RQ_{tp} \leq Q_{tp}\_s$ then $SV2:=SV2 \cup s$
11:      else $SP2:=SP2 \cup s$; end if
12:  end if
13:end for
```

4.5.2 Built Valid Pairs of s1 and s2

Algorithm 2 builds valid pairs of all service candidates, which fit into slot *s1* and *s2*. A pair *(s1,s2)* with $s1 \in SV1$ and $s2 \in SV2$ is valid, if both their IO and PE matching type are Exact or PlugIn (see section 4.4.2.1).

Algorithm 2. Valid Pairs of *s1* and *s2*

```
1: $WF:=\emptyset$
2: for all $s1 \in SV1$ do
3:   for all $s2 \in SV2$ do
4:     if $dest\_s1 \sqsubseteq origin\_s2$ then
5:       $time:= Q_{et}\_s1+d_{m\_s1,\, m\_s2}(w_{dest}\_s1)+ Q_{et}\_s2$
6:       $reliability:=Q_{re}\_s1 \cdot Q_{re}\_s2$
7:       if $(time \leq RQ_{et}$ and $reliability \geq RQ_{re})$ then
8:         $wf:=(s1,s2)$
9:         $type\_wf:=sequence$
10:        $Q_{co}\_wf:= Q_{co}\_s1+c_{m\_s1,\, m\_s2}(w_{dest}\_s1)+Q_{co}\_s2$
11:        $Q_{et}\_wf):=time$
12:        $Q_{tp}\_wf:=min(Q_{tp}\_s1,Q_{tp}\_s2)$
13:        $Q_{re}\_wf:=reliability$
14:        $WF:=WF \cup wf$
15:      end if
16:    end if
17:  end for
18:end for
```

IO was already checked in the preceding step. PE relates to matching of *s1*'s destination and *s2*'s source (line 4). Each valid pair is regarded as workflow $wf_i=(s1,s2)$ (line 8) and stored in the return set of workflows *WF* (line 14). QoS of *s1* and *s2* are aggregated as follows: adding execution time including transshipment delay (line 5), multiplying reliability (line 6), adding costs including transshipment costs (line 10), and selecting the minimum of throughput (line 12).

4.5.3 Check Multi-Merge

Potential candidates for multi-merge were already collected in *SP1* (respectively *SP2)*. If two or more candidates with the same origin and destination (Algorithm 3, line 2/3) exist, the following heuristic is applied: a multi-merge contains at least 2 and at most 3 services (line 4). Whereas this prevents optimality it reduces the complexity (otherwise power set of *SP1* per OD-pair). The rationale is to reflect practice, which is often reluctant to splitting freight extensively and re-joining it at an intermodal node.

If the multi-merge meets or exceeds the required throughput (line 8), we add a parallel workflow to *WF1* (line 13), with respectively aggregated QoS (line 11/12). The algorithm for *s2* is very similar, by adding workflows to *WF2* (omitted due to page limitation).

Algorithm 3. Check Multi-Merge for *s1*

```
1: $WF1:=\emptyset$
2: store OD-pairs of $SP1$ in $OD1$
3: if $OD1 \neq \emptyset$ then
4:   $D:=\mathcal{P}'(OD1)$, with $\mathcal{P}'(OD1)=\{U \subseteq X: 2 \leq |U| \leq 3\}$
5:   // D is the power set of cardinality of 2 and 3
6:   for all $d \in D$
7:     $tp:=\sum_{n=1}^{|d|} Q_{tp}\_d$
8:     if $RQ_{tp} \leq tp$ then
9:       $wf:=d$
10:      $type\_wf:=merge$
11:      $Q_{co}\_wf:=\sum_{n=1}^{|d|} Q_{co}\_d$ ;  $Q_{et}\_wf:=max(Q_{et}\_d)$
12:      $Q_{tp}\_wf:= tp$                ;  $Q_{re}\_wf:=min(Q_{re}\_d)$
13:      $WF1 :=WF1 \cup wf$
14:    end if
15:  end for
16:end if
```

4.5.4 Check Loop

A loop pattern distributes load on iterations of the same service. These services are rather fast, while delivering a rather small amount of freight (throughput); hence we look for services with execution time half or less than required, and throughput two times or more higher than required (Algorithm 4, line 3).

Algorithm 4. Check Loop for *s1*

```
1: $SL:=\emptyset$
2: for all $sp1 \in SP1$
3:   if ($RQ_{et} \geq 2 \cdot Q_{et}\_sp1$ and $RQ_{tp} \geq 2 \cdot Q_{tp}\_sp1)$ then
4:     $SL:=SL \cup s$
5: end for
6: for all $sl \in SL$
7:   $k:=RQ_{tp}$ DIV $Q_{tp}\_sl$
8:   if $(RQ_{tp}$ MOD $Q_{tp}\_sl)>0$ then $k:=k+1$
9:   if $RQ_{et} \geq (k \cdot Q_{et}\_sl)$ then
10:    $wf:=(sl_{i,1},\ldots,sl_{i,2},\ldots sl_{i,k})$
11:    $type\_wf:=loop$
12:    $Q_{co}\_wf:=k \cdot Q_{co}\_sl$ ;  $Q_{et}\_wf:=k \cdot Q_{et}\_sl$
13:    $Q_{tp}\_wf:=k \cdot Q_{tp}\_sl$ ;  $Q_{re}\_wf:=Q_{re}\_sl$
```

```
14:     $WF1 := WF1 \cup wf$
15:   end if
16: end for
```

If a loop candidate is found (line 3), we determine the number of iterations *k* to meet the required load (line 7/8) and add a loop workflow to *WF1* (line 16). The QoS aggregation formulae are specific to loop (line 12/13). The algorithm for *s2* is similar, but skipped because of limited space.

4.5.5 Creating and Ranking Service Compositions

The next step is creating service compositions *WFC* by referring to sets *WF*, *WF1*, and *WF2*. We try to add three categories of workflows by:

- Replacing *s1* in *WF* by *WF1* (Algorithm 5). It causes delays (line 5) and costs (line 8) at the transfer node.
- Replacing *s2* in *WF* by *WF2*. It causes also delays and costs for transshipment. In case of loop, these are added *k*-times.
- Pairing *WF1* and *WF2*. Algorithm 6 determines delays and costs depending on workflow type of *WF2* (line 6 and 9).

Algorithm 5. Replace *s1* by *WF1*

```
1:  if $WF1 \neq \varnothing$ then
2:    for all $wf \in WF$
3:      for all $wf1 \in WF1$
4:        $wfc := (wf1, wf\_s2)$;    $type\_wfc := sequence$
5:        $Q_{et}\_wfc := Q_{et}\_wf1 + d_{m\_wf1,m\_s2}(w_{dest}\_wf1) + Q_{et}\_wf\_s2$
6:        $Q_{re}\_wfc := Q_{re}\_wf1 \cdot Q_{re}\_wf\_s2$
7:        if $(Q_{et}\_wfc \leq RQ_{et}$ and $Q_{re}\_wfc \geq RQ_{re})$ then
8:          $Q_{co}\_wfc := Q_{co}\_wf1 + c_{m\_wf1,m\_s2}(w_{dest}\_wf1) + Q_{co}\_wf\_s2$
9:          $Q_{tp}\_wfc := min(Q_{tp}\_wf1, Q_{tp}\_wf\_s2)$
10:         $WFC := WFC \cup wfc$
11:       end if
12:     end for
13:   end for
14: end if
```

Algorithm 6. Pair *WF1* and *WF2*

```
1:  if $(WF1 \neq \varnothing$ and $WF2 \neq \varnothing)$ then
2:    for all $wf1 \in WF1$
3:      for all $wf2 \in WF2$
4:        if $dest\_wf1 \sqsubseteq origin\_wf2$ then
5:          $wfc := (wf1, wf2)$;    $type\_wfc := sequence$
6:          if $type\_wf2 = merge$ then
7:            $Q_{et}\_wfc := Q_{et}\_wf1 + d_{m\_wf1,m\_wf2}(w_{dest}\_wf1) + Q_{et}\_wf2$
8:            $Q_{co}\_wfc := Q_{co}\_wf1 + c_{m\_wf1,m\_wf2}(w_{dest}\_wf1) + Q_{co}\_wf2$
9:          elseif $type\_wf2 = loop$ then
10:           $Q_{et}\_wfc := Q_{et}\_wf1 + k\_wf2 \cdot d_{m\_wf1,m\_wf2}(w_{dest}\_wf1)$
11:                       $+ Q_{et}\_wf2$
12:           $Q_{co}\_wfc := Q_{co}\_wf1 + k\_wf2 \cdot c_{m\_wf1,m\_wf2}(w_{dest}\_wf1)$
13:                       $+ Q_{co}\_wf2$
14:         end if
15:       end if
16:       $Q_{re}\_wfc := Q_{re}\_wf1 \cdot Q_{re}\_wf2$
17:       if $(Q_{et}\_wfc \leq RQ_{et}$ and $Q_{re}\_wfc \geq RQ_{re})$ then
18:         $Q_{tp}\_wfc := Q_{tp}\_wf1 + Q_{tp}\_wf2$
19:         $WFC := WFC \cup wfc$
20:       end if
21:     end for
22:   end for
23: end if
```

The final step is ranking the compositions in *WF*∪*WFC* by costs. The algorithm determines all valid compositions, except for the heuristic contained in Algorithm 3.

4.5.6 Modifications for Connected Hub Template

The current algorithm needs to be modified for the connected hub template as follows: (1) determine candidates for three slots, i.e., pre-carriage as *s1*, main carriage as *s2*, and on-carriage as *s3*, (2) pairing of *S2* and *S3*, (3) check multi-merge and loop also for *s3*, and (4) create compositions based on *WF*, *WF1*, *WF2*, and *WF3*.

5. EVALUATION

This section provides an evaluation of our proposal by conducting simulation experiments, reporting its results and discussing the findings as well as the implications and limitations.

5.1 Experimental Setup

The main objective of the simulation experiment is to study the algorithms performance. A second objective is to instantiate the modeling approach with realistic data.

Transport system and services: We consider an ITS across Germany, which is segmented into three regions. Each region contains locations, being connected by relations. Freight must be routed from origins in region #1 to destinations in region #3 via a hub in region #2 (hub template). LSPs offer transport services for these relations. We reuse the United Nations Code for Trade and Transport Locations (UN/LOCODE) [20], which classifies each location by general modes, for setting up a realistic *ITS*. We add delays and costs for modal changes to each location. We define a light-weight ontology for transport units *TU* comprising seven concepts for containers and three concepts for palettes, allowing to transport palettes by (a subset of) containers. Table 6 gives the figures for *ITS* and *TSF*.

Table 6. Basic experimental design of *ITS* and *TSF*

Component	Instantiation
$\lvert N \rvert$	150, snapshot from UN/LOCODE
$\lvert R \rvert$	25,320
C, D	[10;20], uniform distribution
S	For all $s \in S$: $\lvert MA(s) \rvert = 1$
Q_{et}, Q_{co}, Q_{tp}	[50;150], uniform distribution
Q_{av}	Lower and upper bound of a discretized time period of interest; uniform distribution of bounds
Q_{re}	[0.9;1], uniform distribution

Routing requests: We consider three different types of requests. Type A with high reliability Q_{re}, which can most probably be met by merges; Type B with high throughput Q_{tp} placing emphasis on merges; and Type C with short execution time Q_{et} (Table 7).

Table 7. Types of freight routing requests

Component	Type A	Type B	Type C
n_{origin}	Uniformly distributed within region #1		
n_{dest}	Uniformly distributed within region #3		
RQ_{et}	400	250	**150**
RQ_{tp}	80	**300**	80
RQ_{av}	20	20	20
RQ_{re}	**0.98**	0.95	0.95

Variations: The set of experiments covers two variations. The number of services per relation is sr with $sr=5,10,15,20,25$. The number of considered nodes is $|N'|$ with $N' \subseteq N$ and $|N'|=30,60,90,120,150$. The number of services connecting region #1 and #2 is then given by $sr \cdot (|N'|/3) \cdot (|N'|/3)$ (same as for region #2 and #3).

Prototype system: We implemented a prototype system using Visual Basic for Applications 2003; it stores all data about *ITS*, *TSF*, and workflows in a SQL database (further specification: Intel Centrino Core Duo T2500 CPU, 2 GB RAM, Windows XP Professional). We avoided the potential bottleneck of a OWL reasoner for the domain ontology by re-implementing the light-weight reasoning directly within the database-oriented system. The reason is to concentrate on the core algorithm and to avoid limitations of current SOC software packages.

5.2 Results

Each experiment consists of 100 requests per request type. As a performance metric we use the CPU time per request in seconds. We also measure the returned workflows by calculating mean, min, max, and standard deviation for the following sets: *WF*, multi-merges in *WF1*, loops in *WF1*, multi-merges in *WF2*, loops in *WF2*, and *WFC* (returned by algorithm 1 to 6).

Varying services per relation: Table 8 and Figure 2 show the CPU time per request for a 30 nodes *ITS*. The total number of services increases from 1,000 *(sr=5)* to 5,000 *(sr=25)*. The results indicate a dependency on the request type: type A requires almost linear computational time, whereas B and C suggest an exponential complexity.

Table 8: CPU time (sec) with varying services per relation

$\|S\|$ per r	5	10	15	20	25
Type A	0.09	0.15	0.22	0.31	0.53
Type B	0.18	0.52	1.51	4.29	10.48
Type C	0.12	0.38	1.01	2.52	4.98

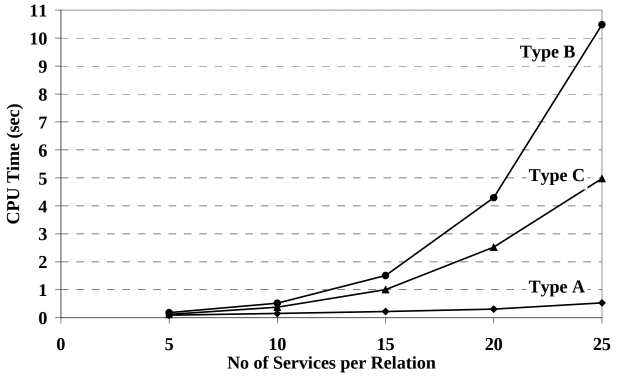


Figure 2. Performance with varying services per relation.

Table 9 gives the numbers of resulting workflows (mean) for $sr=10$ and $sr=25$. Multi-merges are listed as MM.

Varying number of nodes: Table 10 and Figure 3 show the CPU time per request for $sr=15$. Because of the interwoven *ITS* (i.e., every node in region #2 has 10 to 50 successors respectively predecessors), the number of services increases very much. For $|N'|=150$ it amounts to $2 \cdot 15 \cdot 150/3 \cdot 150/3=75,000$ services.

Table 9. Workflows (mean) for 10 and 25 services per relation

No	Type	WF	MM WF1	MM WF2	Loop WF1	Loop WF2	Pair WF1/2
10	A	1.42	0.22	0.14	0.18	0.24	0.00
	B	0.00	9.86	9.32	0.00	0.00	0.20
	C	9.51	1.48	1.69	0.02	0.06	0.00
25	A	9.26	1.31	2.46	4.55	12.79	0.00
	B	0.00	175.86	136.98	0.00	0.00	209.70
	C	57.68	6.74	10.92	11.59	10.94	0.00

Table 10. CPU time (sec) with varying number of nodes

$\|N'\|$	30	60	90	120	150
Type A	0.22	0.70	1.50	2.60	3.91
Type B	1.51	3.29	5.01	7.63	10.76
Type C	1.01	3.02	7.03	12.32	19.57

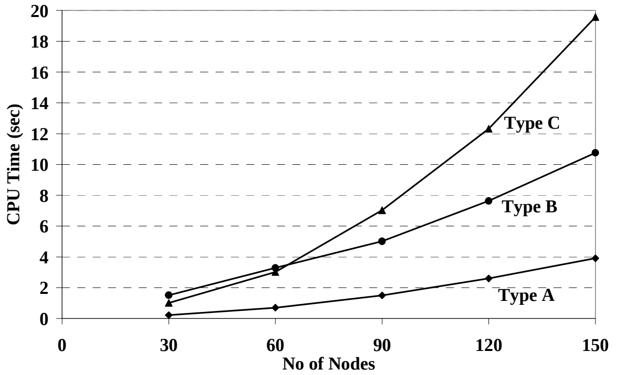


Figure 3. Performance with varying number of nodes.

5.3 Discussion

5.3.1 Simulation Experiments

The proposed algorithm determines freight routes effectively. Its complexity depends to a high extend on the routing request as well as number of nodes and services. Even the largest setup of 150 nodes and 75,000 services requires less than 20 seconds of CPU time.

As Table 9 shows, the resulting workflows differ a lot: A returns *WF*, multi-merges and loops, and C yields *WF* and multi-merges. B can not be fulfilled by *WF*, but results in a non-linear increase of merges, loops, and in particular combinations of such slots, thus pairs of *WF1* and *WF2*. The workflow sets grow due to the algorithm's rationale to generate *all* valid routes (except for the heuristic for multi-merges contained in Algorithm 3). This approach allows selecting from these routes and considering trade-offs, e.g., between cost Q_{co}, time Q_{et}, and reliability Q_{re}.

The evaluation by simulation does not determine complexity formally, but gives indications of complexity. For instance, an analysis of Table 10 allows the following estimation of exponents, if we limit these to only one variable: 1.7 for A, 1.1 for B, and 1.8 for C. These results suggest that the number of nodes $|N'|$ has a smaller influence on computational time than services per relation rs. It should be noted, though, that the algorithm's performance is influenced by, e.g., number of modes, and size and richness of domain ontology, which are not tested in this paper.

An easy way to improving scalability is adding a cost restriction RQ_{co} to the routing request and amending algorithm 1 to 6. Though this measure would only exclude expensive service compositions, whereas fail if the restriction is too weak. A more promising modification would be to set a maximum for the number of workflows in *WF1* and *WF2*, which reduces the number of subsequent combinations to be tested.

5.3.2 Modeling Approach

The service-oriented modeling approach adopts and interprets the IOPE and QoS models, which both were not developed for electronic representations of logistics services. The question is to which extent this approach is capable to representing barriers of intermodality. We answer this question by comparing our proposal to existing research (as reviewed in section 2.1). We refer to seven criteria related to intermodality as shown in Table 11. Our approach fulfills five out seven criteria.

A unique characteristic is the consideration of an arbitrary number of interrelated transport unit types; this criterion is covered due to a semantic service description and IO constraints, which relate service properties to a domain ontology including transport units. The templates used for composing services limits, however, the number of relations to 3. Another important difference is the consideration of competitive offerings by LSP. In particular, we allow parallel and iterative splits of freight. All other approaches determine routes directly from relations, whereas the selection of a concrete LSP per relation is delegated to a subsequent decision.

Table 11: Comparison of freight routing approaches

Criteria	Freight routing approach					
	[15]	**[3]**	**[4]**	**[7]**	**[26]**	**Pro-posal**
No of transport unit types	1	1	1	1	1	Any
No of modes	3	Any	3	Any	Any	Any
No of relations per route	5	Any	5	Any	Any	3
Competition between LSPs	No	No	No	No	No	Yes
Transshipment delays	Yes	Yes	Yes	No	Yes	Yes
Transshipment costs	No	Yes	Yes	No	No	Yes
Time windows for modes	No	No	No	Yes	No	No

5.3.3 Implications

Usage scenario: This research contributes to flexibility of intermodal transport systems by making service offerings at local nodes visible and accessible to other parties. The usage scenario concerns a multi-tier supply chain of (1) customers/shippers, (2) third-party logistics service providers (3PL) offering intermodal transport, and (3) second-party logistics service providers (2PL) offering transport on a subset of relations, most often confined to one mode of transport (e.g., road transport by trucking companies, air transport by air cargo shippers, etc.).

The adoption of loosely coupled services pays respect to fragmentation of intermodal transport systems, thus there exist actually single- or dual-mode subsystems governed by local actors [8][26]. Due to division of labor, LSP that offer intermodal end-to-end transports rely essentially on local actors and outsource sub-transports to them [15].

Two use cases can be distinguished: First, a shipper searches for an intermodal transport by retrieving service offerings from both 2PL and 3PL. Second, a 3PL coordinates intermodal transports in an *open* transport system. Openness describes that the number of 2PL is more flexible than those in transport systems tailored for a specific type of freight, in particular those that exist for courier, express, and parcel (CEP) services (such as FedEx, UPS). In both cases, service offerings need to be made available in some form of an electronic marketplace. This requirement is addressed by service description and registration as follows.

Service description: Service offerings need to be described formally and annotated according to a shared domain ontology. Since these specifications are nowadays not used, our research suggests converting current data of transport management systems (TMS) into the IOPE-based service model. The experience made during conducting the simulation experiments is that the proposed models can be instantiated effectively by referring to logistics data such as locations, costs, delays, and that a light-weight domain ontology can be derived rather quickly.

Service registration: An implementation must acknowledge that LSPs are not willing to disclosure all service parameters, in particular tariff schemes. Today's TMS often contain only standard schemes or no price information at all; the returned route is then used for submitting a request for quotation (RfQ) to the selected LSP. Our proposal fits into this picture, i.e., by replacing concrete cost Q_{co} by defaults. The service model as well as the composition algorithm do not include a dedicated pricing model, because this is still subject to the TMS. Therefore, service compositions returned by freight routing represent only a stage of an overarching business process.

Information Systems Research: From the academic perspective, a direct implication of our research is that SOC's concepts and formal models of service composition can effectively be used for intermodal freight routing, thus solving a coordination problem in logistics. This usage was possible by interpreting software-based services as services in economics (here: limited to transport operations), unlike the conventional interpretation as elements of a software architecture. Beyond the particular problem of intermodal freight routing, this research is part of general studies of coordinating activities in logistics based on loosely coupled software services and the SOC technology stack. The initial results suggest that this modeling approach provides both the required expressivity to covering domain constraints and mechanisms for matchmaking of logistics demand and supply.

5.3.4 Limitations

This research has some limitations. The current prototype is not ye a "true" SOC prototype, which would contain service descriptions in OWL-S, workflow specifications in WS-BPEL, and its domain ontology in OWL. The reason is that it should first validate the overall modeling approach and the composition algorithm. We plan to extend the prototype system to process and generate semantic services descriptions in OWL-S. In addition, more comprehensive experiments are needed, with regard to complexity analysis as well as practical adoption by integrating real-world data from TMS. This work takes place in collaboration with two logistics software companies, which will provide real-world data for enhancing the simulation experiments and end-users for a realistic evaluation.

The current research does not provide a logistics ontology yet. Future work is required to enriching the semantic foundation of *T* and providing the necessary reasoning over semantic service descriptions as defined in semantic links.

6. CONCLUSION

This paper proposed a SOC perspective to solving an operational problem in intermodal transport logistics and developed a service composition algorithm. The contribution is service-oriented freight routing that which takes into account barriers between relations. Experimental runs show evidence of the validity and usefulness of our approach.

7. ACKNOWLEDGMENTS

The work presented in this paper was partly funded by the German Federal Ministry of Education and Research under the project InterLogGrid (BMBF 01IG09010E).

8. REFERENCES

[1] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P. 2003. Workflow patterns. *Distrib. Parallel Dat.* 14, 3 (Jul. 2003), 5-51.

[2] Blau, B., van Dinther, C., Conte, T., Xu, Y., and Weinhardt, C. 2009. How to coordinate value generation in service networks – a mechanism design approach. *Bus. Inform. Syst. Eng.* 1, 5 (Oct. 2009), 343-356.

[3] Boardman, B.S., Malstrom, E.M., Butler, D.P., and Cole, M.H. 1997. Computer assisted routing of intermodal shipments. *Comput. Ind. Eng.* 33, 1-2, 311-314.

[4] Bookbinder, J.H. and Fox, N.S. 1998. Intermodal routing of Canada-Mexico shipments under NAFTA. *Transp. Res. E-Log.* 34, 4, 289-303.

[5] Caris, A., Macharisb, C., and Janssens, G.K. 2008. Planning Problems in Intermodal Freight Transport: Accomplishments and Prospects. *Transp. Plan. Techn.* 31, 3 (Jun. 2008), 277-302.

[6] Casati, F. and Shan, M.-C. 2001. Dynamic and adaptive composition of e-services. *Inform. Syst.* 26, 3 (May 2001), 143-163.

[7] Chang, T.-S. 2008. Best routes selection in international intermodal networks. *Comput. Oper. Res.* 35, 9, 2877-2891.

[8] European Commission. 2001. *European Transport Policy for 2010: time to decide*. White Paper, Luxembourg.

[9] Giannopoulos, G.A. 2004. The application of information and communication technologies in transport. *Eur. J. Oper. Res.* 152, 302-320.

[10] Iyer, B., Freedman, J., Gaynor, M., and Wyner, G. 2003. Web services: enabling dynamic business networks. *Commun. AIS.* 11, Article 30 (2003).

[11] Kunkel, M.; Doppstadt, C, and Schwind, M. 2009. Open Service-Oriented Computing for Logistics: A Case in Courier, Express and Parcel Network. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*. LNCS 6275, 134-144.

[12] Lécué, F. and Delteil, A. 2007. Making the difference in semantic web service composition. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (Vancouver, Canada, Jul. 22 - 27, 2007). AAAI 2007, 1383-1388.

[13] Milanovic, M. and Malek, M. 2004. Current solutions for Web service composition. *IEEE Internet Comput.* 8, 6, 51-59.

[14] Min, H. 1991. International intermodal choices via chance-constrained goal programming. *Transp. Res. A-Pol.* 25, 6 (Nov. 1991), 351-362.

[15] Panayides, P.M. 2002. Economic organization of intermodal transport. *Transp. Rev.* 22, 4 (Oct. 2002), 401-414.

[16] Papazoglou, M.P., Traverso, P., Dustdar, S., and Leymann, F 2007. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Comput.* 40, 11 (Nov. 2007), 38-45.

[17] O'Sullivan, J., Edmond, D., and ter Hofstede, A.H.M. 2002. What's in a Service? *Distr. Parallel Dat.* 12, 2-3 (Sep. 2002), 117-133.

[18] Röglinger, M. 2009. Verification of Web Service Compositions: An Operationalization of Correctness and a Requirements Framework for Service-oriented Modeling Techniques. *Bus. Inform. Syst. Eng.* 1, 6 (Dec. 2009), 429-437.

[19] ten Teije, A., van Harmelen, F., and Wielinga, B. 2004. Configuration of Web Services as Parametric Design. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management* (Whittlebury Hall, UK, Oct. 05 - 08, 2004). IKAW 2004, 321-336.

[20] UNECE. 2010. *United Nations Code for Trade and Transport Locations*. URI=http://www.unece.org/cefact/locode.

[21] W3C. 2004. *OWL-S: Semantic Markup for Web Services.* URI=http://www.w3.org/Submission/OWL-S.

[22] Woxenius, J. 2007. A generic framework for transport network designs: applications and treatment in intermodal freight transport literature. *Transp. Rev.* 27, 6 (Nov. 2007), 733-749.

[23] Xiang, Y., Zhang, S., Shen, Y., Shi, M. 2009. Pattern-Oriented Workflow Generation and Optimization. *J. Univers. Comput. Sci.* 15, 9, 1924-1944.

[24] Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and Chang, H. 2004. QoS-Aware Middleware for Web Services Composition. *IEEE T. Software Eng.* 30, 5 (May 2004), 311-327.

[25] Ziliaskopoulos, A. and Wardell, W. 2000. An intermodal optimum path algorithm for multimodal networks with dynamic arc travel times and switching delays. *Europ. J. Oper. Res.* 125, 3 (Sep. 2000), 486-502.

[26] Zografos, K.G. and Regan, A.C. 2004. Current Challenges for Intermodal Freight Transport and Logistics in Europe and the United States. *Transp. Res. Rec.* 1873, 70-78.