Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 72 (2023) 1715–1722

Transportation Research Procedia

www.elsevier.com/locate/procedia

Transport Research Arena (TRA) Conference

Modelling shared logistics resources in loosely coupled software architectures

Florian Peters[a] , Sabine Limbourg[a1]

[a]*University of Liège, Liège, Belgium*

Abstract

This research is part of the City Line project, a collaborative initiative within Belgium aiming to promote collaboration between various logistics operators, from cargo bikes to trucks, while ensuring low emissions for last-mile deliveries in cities. The objective is to integrate all relevant information regarding shared assets into a digital collaborative logistics platform managed by a third-party logistics provider acting as a central decision-maker coordinating transportation activities. First, we describe how the digital platform receives the services offered by the logistics operators and the delivery requests. Then, we propose an incremental method to assign the transport demand to the best route according to environmental and economic factors.

Keywords: collaborative logistics; physical internet; digital platform; share resources

1. Introduction

Urban freight transport is an essential component of the functioning of cities. However, it negatively affects the environment causing externalities such as greenhouse gas emissions, air pollution, noise and traffic congestion (Strulak-Wójcikiewicz and Wagner, 2021; Nuzzolo and Comi, 2015). To meet the Global Logistics Sustainability Grand Challenge, Montreuil (2011) proposes his Physical Internet vision. In Montreuil et al. (2013), the Physical Internet is defined as "an open global logistics system founded on physical, digital and operational interconnectivity through encapsulation, interfaces and protocols". The digital interconnectivity guarantees that physical agents can exchange information such as tracking shipments among carriers, the state of demand, offer, flow, and capacity to serve a common customer by maximizing the utilization of their resources.

Horizontal collaboration with competitors provides significant advantages such as saving costs, improving global transport efficiency and extending one's service zone. There are, nonetheless, potential pitfalls when collaboration is

[1]* Corresponding author. Tel.: +32-4-366-31-81.
E-mail address: sabine.limbourg@uliege.be

10.1016/j.trpro.2023.11.645

involved, including inappropriate partner selection, unequal distribution of costs and benefits or unbalanced power dynamics. Moreover, competitors are, of course, reluctant to share confidential information. Verdonck et al. (2013) show that a cooperation technique can be applied when a freight forwarder acts as a third-party logistics provider that receives transportation requests from shippers and assigns these optimally to collaborating carriers. Indeed, in this manner, the freight forwarder serves as a central decision-maker coordinating transportation activities based on his knowledge of all customer orders, vehicle capacities and transportation costs.

The City Line project, a collaborative initiative funded by the Walloon Region in Belgium, aims to promote collaboration between various logistics operators, from cargo bikes to trucks, while ensuring low emissions for last-mile deliveries in cities. The mechanism of pooling different actors' fleets is a positive disruptor in multimodal logistics and could provide a solution to the Global Logistics Sustainability Grand Challenge. Within this context, the contribution of this paper is to explain how to model logistics resources in loosely coupled software architectures. We also provide a concrete framework to implement collaborative logistics platforms in practice. Additionally, we introduce a completed system deployed in a real logistics setting.

In the next section, the problem of logistics resource sharing is described. Then, Section 3 presents the digital collaborative logistics platform, and Section 4 the incremental assignment method. In the final section, conclusions and future research opportunities are presented.

2. Problem description

Let us assume a delivery request originating from a low emission zone (LEZ) A and having as a destination a different LEZ B. Fig. 1 illustrates this situation. The transport request can be satisfied using three service providers, as represented in Fig. 2. However, these fragmented supply chain networks can lead to sub-optimal planning. Indeed, as each logistics operator only concerns transport orders in its activity zones, orders spanning multiple activity zones cannot be served. Furthermore, such a network topology leads to an inability to consolidate shipments to efficiently achieve greater economies of scale. As a result, aggregation of orders is not feasible or requires an additional investment from the requester to contact multiple transport providers and combine their services. Fig. 2 illustrates this sub-optimal behaviour.

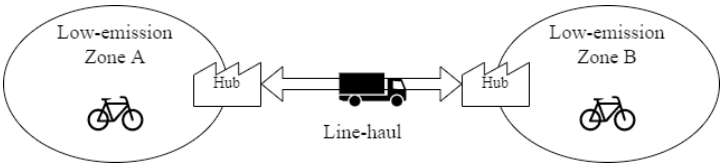

Fig. 1. Diagram of an intercity delivery network.

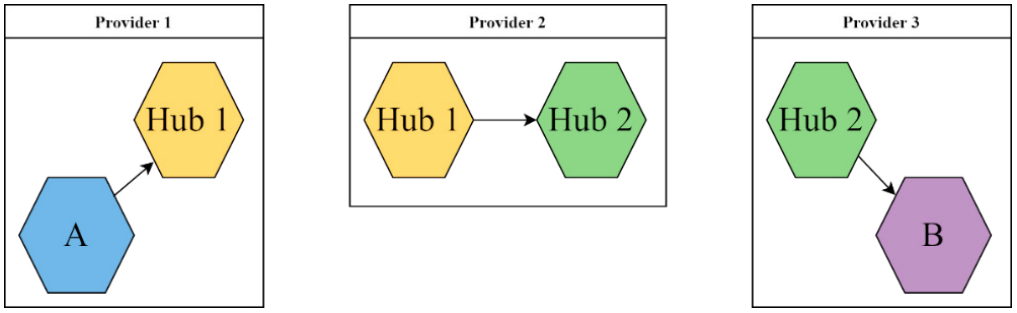

Fig. 2. Abstract transport network comprises three separate providers, each supplying their zones of activity. Coloured hexagons represent delimited geographical areas, and directed edges indicate a transport service from one zone to another. In such a configuration, the network is unable to fulfil transport orders spanning multiple zones of activity due to their fragmented nature. For example, this network configuration cannot directly fulfil an order originating from zone A and arriving in zone B.

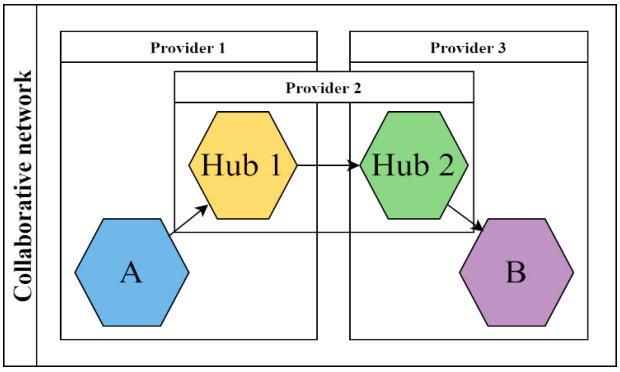

Fig. 3. Example of a collaborative network comprised of three different providers. By pooling services across multiple providers, the collaborative network can handle requests that would be impossible to fulfil if each provider worked independently.

Transport providers struggle to find adequate transport flow assignment strategies that satisfy their customers and guarantee long-term profits. Traditional transport providers are often unwilling to service city centres which can be explained by a higher proportion of missed appointments and other complications in urban areas than in non-urban ones (Dablanc, 2007). Other times, it is simply impossible to reach areas of certain cities due to new LEZs legislation. An alternative is to create a parent network that can dispatch shipments and delivery requests to transport providers cost-efficiently. Each provider benefits from a greater market reach by pooling logistics resources: requests that were impossible to handle can now be sent across multiple transport services. Moreover, the third-party entity knows more about the entire network through data sharing with its partner providers. This knowledge creates a clearer picture of what is and is not feasible within the entire network, allowing for more efficient transport. By acting from a bird's eye view, the parent network can aggregate shipments from various requests onto the same transport services, generating more significant economies of scale. This advantage can only be achieved if the network is aware of all the transport flows, leading to the definition of a collaborative logistics network.

We define a collaborative logistics network as a transport network composed of multiple transport providers and a third-party master provider responsible for dispatching delivery requests to its constituent providers. The relationship between the master and partner providers is formalized by an agreement known as the framework contract, which specifies the nature of the pooled logistics resources, transport tariff, time, location and capacity constraints, and minimum transport volumes. The typical workflow for a transport request in this network configuration is summarized as follows: first, handle the first-mile pick-up in LEZ A using a cargo bike logistics operator that will then deliver the shipment to a common hub belonging to the network (Hub 1). Then, handle the transport from this hub in LEZ A to a final hub (Hub2) in LEZ B of a different city via line-haul transportation. Finally, the last-mile delivery can be handled by soliciting a bike logistics operator whose activity zone includes LEZ B. Fig. 3 illustrates this collaborative configuration.

City Line aims to solve a real case of a fragmented supply chain network in Belgium. It began as a collaboration between multiple transport providers, end-users, IT professionals and research centres which come together to implement a collaborative logistics solution that is digital and founded on realistic principles. In this project, the collaborative network relies on services. A service connects areas (ZIP codes or depots) considering time windows, weight, emission and size constraints. Each operator offers services via a framework contract, and each offered service is agreed with a tariff. The digital collaborative logistics platform lists the eligible operators who can handle this type of delivery and with available resources. Then an incremental method assigns the transport demand to the best route according to environmental and economic factors. Transport services are the central components of a collaborative network. Without a clear definition of services, generating an abstract transportation network to test transport scenarios and delivery rounds becomes impossible. Moreover, services are essential when deciding if a given sequence of providers can fulfil the overall delivery request: services determine each route's capacity and time constraints. Therefore, services are the key digital resources when modelling a collaborative logistics network and require the utmost care in their definition.

A collaborative approach to handling transport flows has several benefits. First, it increases the market reach of each transport provider by allowing multimodal flows that would otherwise be impossible to treat on their own. Second, it allows for more significant economies of scale through aggregating those transport flows: there are more

shipments on inter-hub lines, reducing environmental and economic costs per shipment. In practice, this cooperative strategy can be attained by introducing a third-party entity acting as the bridge between all providers. An appropriate choice for this entity is an automated digital platform as it allows for faster response times and an ability to generate potential routes for deliveries quickly. The following sections detail how to build such a digital platform from the ground up.

3. Digital collaborative logistics platform

In order to be efficient in a real context, a digital logistics platform must be able to reason accurately on actual logistics resources. Such a feat can be accomplished via a digital simulation approach; this digital model can then be probed to infer the best possible decisions to maximize or minimize an objective function. In this context, the platform should be able to reason about the following key logistics entities: *Transport providers, Requesters, Transport services* and *Shipments*.

A complex modern software solution is typically developed using a modular approach: the complete software package can be subdivided into multiple independent components called modules. Furthermore, this modular approach follows the design principles of loosely coupled systems. In this paper, a loosely coupled system is defined as in Kaye (2003). A loosely coupled system is a system in which:

- Components are weakly associated (have a breakable relationship) with each other, so changes in one component least affect the existence or performance of another component.
- Each of its components has or uses little or no knowledge of the definitions of other separate components.

In order to understand which modules are required, the main data flows to and from the platform should be identified. Fig. 4 provides a high-level overview of the typical workflow when working with a digital collaborative platform. The first step involves the transport provider defining the terms of the framework contract. This includes information such as pick-up and drop-off zones, types of vehicles used, constraints on the shipment types and tariffs. In step 2, an end-user can ask for a delivery request by specifying the pick-up and drop-off zones, the time constraints as well as additional information such as the dimensions, weight and number of shipments. The platform then confronts the request with its abstract transportation network model and sends a response in step 3. The end-user is immediately notified if their delivery is feasible or not. Finally, step 4 is composed of multiple operations. Since the delivery request will most likely be fulfilled using a sequence of services from different operators, the platform decomposes the delivery into multiple pick-up and drop-off orders to its partner transport providers. These communications are made using GS1 standards[21], which provide a clear structure for transport order data exchanges between providers, requesters and the platform. This would make it possible to interface the platform with proprietary software with minimal software development investment. The inner workings of this digital platform are described at length in the following sections.

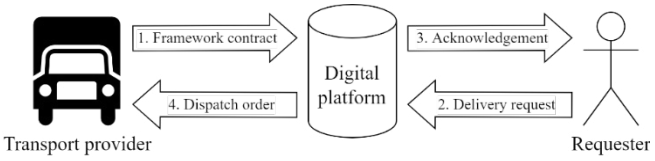

Fig. 4. Overview of data flow when operating a digital collaborative logistics platform.

[21] https://www.gs1.org/standards.

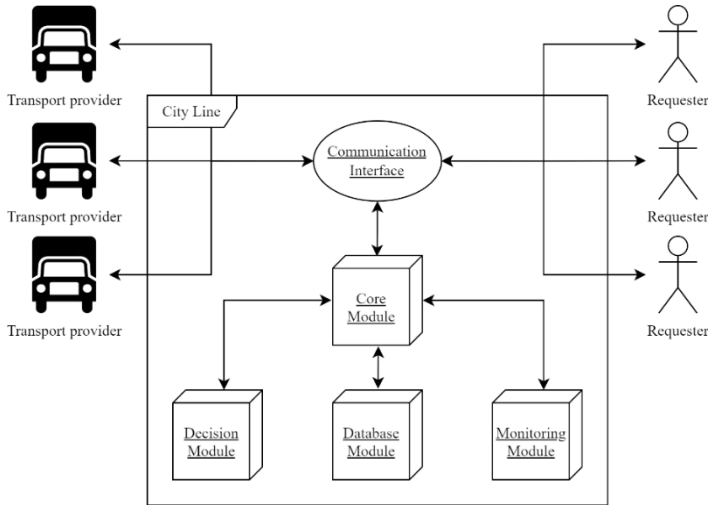

Fig. 5. Internal data flow of the City Line platform.

The nature of the City Line project is such that each member is attributed a specific work package with well-defined scopes and responsibilities. As a result, the completed digital platform is a complex system of interconnected software packages developed independently of one another, an example of which is shown in Fig. 5. This falls within the framework of loosely coupled systems. There are several benefits to this approach:

- Each module can be developed in parallel, reducing overall development time.
- Packages can be developed using the preferred tools of the assigned project member.
- It creates a modular system whose components can be swapped easily to provide additional functionality or improve performance over time.

Nevertheless, this loosely coupled system design also involves several key challenges: strong emphasis needs to be made on data exchanges, as they must remain consistent across multiple modules at once, and each module's behaviour and interfaces must be clearly documented to allow easy integration within the larger system. To face these issues, we focus on describing the decision module from a high-level perspective. This module is not accessible directly from another module: an additional communication layer known as a Representational State Transfer Application Programming Interface (REST API) is developed and constitutes the main mode of communication between modules. REST API is a term coined by Fielding (2000), which identifies programming interfaces that do not store the calling program's state between usages. In other words, the caller is responsible for sending its state to the API for the system to compute an answer. In practice, this is typically implemented in the form of an HTTP server which is the approach taken for the City Line project.

To represent the structure of the decision module, we choose the Unified modelling Language (UML) presented in Booch et al. (1998) since it provides some degree of standardization and it is compatible with any higher-order object-oriented (or class-based) computing language. Fig. 6 is a class diagram of the inner workings of the decision module, specifying the classes within the system, and their relationships to each other. A digital equivalent mirrors each relevant logistics resource. A class, represented as a rectangle, describes a concept such as a delivery request that may have associated attributes and operations.

Interfaces allow for decoupling digital resources, leading to more flexible software architecture as no single class becomes entirely dependent on a given software implementation. For instance, the *Location* interface provides a common framework to communicate with geographical data without knowing the specific implementation details of the underlying software objects. Similarly, the *ServiceTime* interface provides a framework of communication for data that is temporal. Particularly, it focuses on schedules and estimated fulfilment times of services.

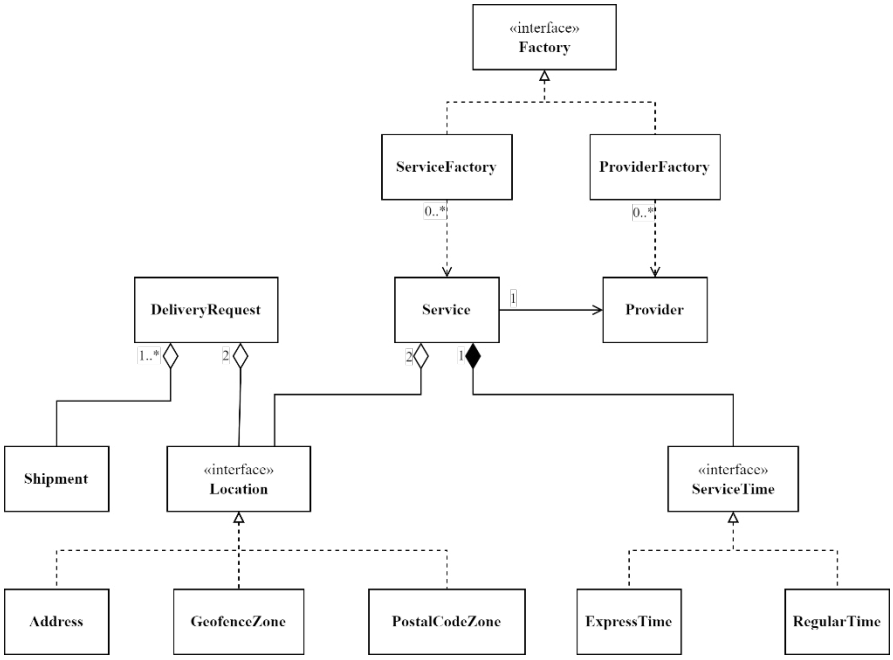

Fig. 6. Class diagram of the software architecture within the decision module.

The key digital resource is the service used to build the abstract network. This is reflected in the class diagram as the *Service* class is linked to almost all other classes. Each *Service* object is created and managed using a *ServiceFactory*, a specific variant of the well-known *Factory* design pattern described by Gamma et al. (1995). The *Factory* is responsible for tracking each *Service* object to retrieve key information when running assignment algorithms. This approach ensures that no *Service* object can be created twice. Furthermore, creating and managing the *Service* is decoupled from other parts of the code since they must go through the *ServiceFactory* class instead.

Let us describe the composition of the *Service* class in more detail. A *Service* is linked to a single *Provider*, responsible for managing and fulfilling the service in accordance with its framework contract. This relationship means that constraints applied to providers directly affect the constraints on services. A *Service* is made of two *Location* objects, one for the origin point and another for the destination. This is shown on the diagram using an aggregation relationship represented as a hollow rhombus with a line. The aggregation relationship means that a *Service* object is made of two *Location* objects that can still exist after the *Service* object is deleted. As mentioned previously, we use an interface to decouple the implementation details of a *Location* object from the *Service*. Thus, we can now define three types of location-related classes: the *Address*, which is a simple postal address with coordinates; the *PostalCodeZone* which includes an entire postal code as the location; and the *GeofenceZone* which defines a specific geographical zone using GeoJSON format proposed by Butler et al. (2016). Each of those specific classes will be used in practice when dealing with different service types. For example, a line-haul service will use two *Address* locations since the service is only concerned with moving freight between two hubs. Finally, a *Service* object works with a *ServiceTime* object via a composition relationship, represented as a black rhombus with a line. The composition means that the *Service* owns the *ServiceTime* object: if the *Service* is deleted, the *ServiceTime* ceases to exist as well. There are two schedule-related classes: the *ExpressTime*, which contains scheduling information when in the context of an express delivery service, and the *RegularTime*, which contains information for regular delivery services.

The digital model also requires information concerning the delivery requests represented in the software architecture using the *DeliveryRequest* class. A request is characterized by a list of *Shipment* objects containing information about the transport items: the weight, dimensions, type of container and the amount of each item. Since a shipment can exist independently of a delivery request, the relationship between the two is an aggregation. Like *Service* objects, *DeliveryRequest* objects are defined by an origin and destination location, hence the aggregation relationship with the *Location* interface. From this structure, we can now develop a REST API that will be accessible from other parts of the platform. To provide flexibility, a CRUD approach as described by Martin J. (1983) is adopted.

The HTTP server supporting the API runs on Python using the Flask library. The main data format for exchanges between modules uses the JSON format since it is lightweight, readable and easy to use. Each class presented earlier possesses an equivalent representation in JSON. Finally, the REST API, being provided as an HTTP server, includes a documentation which allows developers for the other modules to connect their code to the decision module. Fig. 7 represents two screenshots (a) illustrate an example of a *Service* object in JSON and (b) the API's documentation.

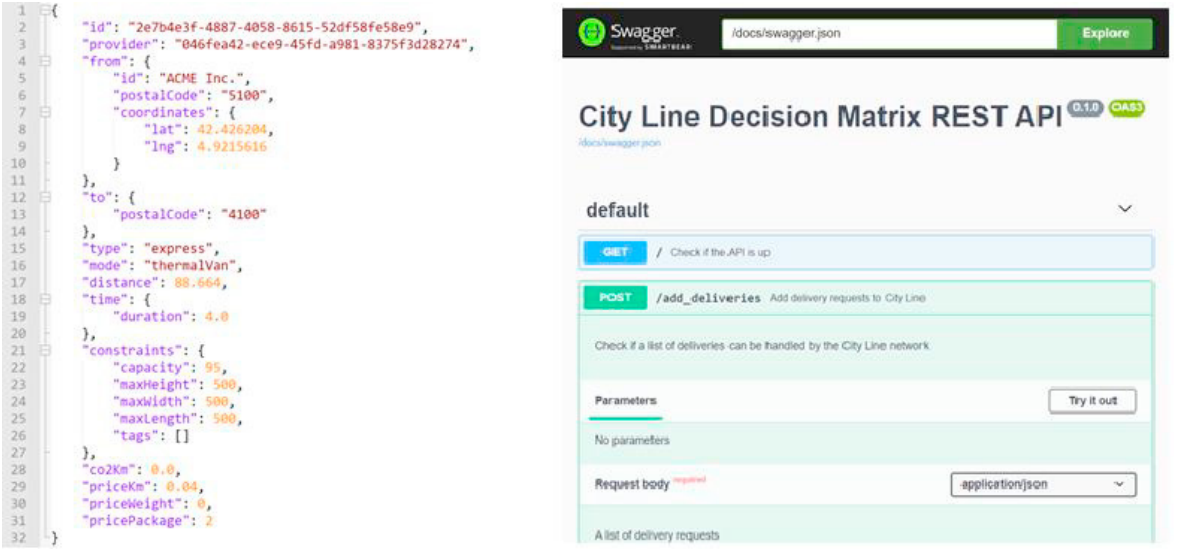

Fig. 7. (a) JSON representation of a *Service* object; (b) Documentation of the API, including sample JSON data and all the endpoints available. Each endpoint is accompanied by a summary and a description of each of its parameters.

- **Incremental assignment**

The incremental assignment is composed of two main steps. The first one generates all feasible paths between each origin-destination pair, as described in Tawfik and Limbourg (2019). It starts by examining all services that originate from the origin of the delivery request. If this service also has the delivery request destination as its destination, then this service is added to the possible routes. Otherwise, the procedure searches for other services to be combined with the previous service to the delivery request destination. All feasible paths must be identified for each delivery request; we set a maximum of three services on a path. A path is considered feasible if it satisfies location, time, and capacity constraints for a given delivery. The second step is to assign each delivery request to a path to minimize both the environmental impacts and costs using a procedure similar to the incremental assignment method depicted in de Dios Ortúzar and Willumsen (2011) with a weighted sum (cost and environmental impacts) attributed to each path. The costs and environmental impacts are estimated using the results obtained in Fraselle et al. (2021). The delivery request is assigned to the path with the lowest weighted sum. The procedure is repeated for the remaining deliveries considered until all deliveries have been assigned to a path or the network cannot accommodate any additional delivery.

This method brings several benefits. First, the implementation can be done using classical graph traversal algorithms such as the algorithm proposed by Dijkstra (1959). This directly translates to short computation times, improving the module's usability. Second, due to the incremental nature of the method, the algorithm can be used in an online fashion, meaning that it does not require all the information to be available to run the algorithm.

- **Conclusion**

In this paper, we model logistics resources in loosely coupled software architectures. All relevant information is integrated into a digital collaborative logistics platform. Its decision module is explained using UML, and the assignment is an incremental one. As modelling logistics resources in loosely coupled software architectures is a recent research topic, several opportunities for future research can be identified. The operational collaboration requires a low commitment level and can be achieved through a digital platform. However, collaborative logistics highlights

the need for specific methods based on game theory approaches to support decision-making at a strategic level, particularly on the framework contract. Additionally, using an online incremental approach can lead to sub-optimal solutions since only a single request is considered at a time; the algorithm might miss a better assignment strategy by slightly changing previously assigned shipments. One possible improvement would be considering multiple deliveries in a batch fashion to reduce the total costs further. Furthermore, using a weighted sum as the objective function requires tuning the weights for each term. What is considered optimal becomes subjective with respect to the chosen weights. Another line of future research could focus on developing more sophisticated heuristics to improve the assignment method.

Acknowledgements

This publication presents the results of research carried out in the framework of the City Line project funded by the Walloon Region, grant number 8294. This City Line project is supported by the cluster Logistics in Wallonia. The authors thank the City Line partners for data sharing.

References

Booch, G., Jacobson, I., & Rumbaugh, J., 1998. Unified Modeling Language, Notation Guide, Version 1.0.

Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., Schaub, T., 2016. RFC 7946: The GeoJSON format. Internet Engineering Task Force (IETF). https://datatracker.ietf.org/doc/html/rfc7946

de Dios Ortúzar, J., & Willumsen, L. G., 2011. Modelling transport. John wiley & sons.

Dablanc, L., 2007. Goods transport in large European cities: difficult to organize, difficult to modernize. Transportation Research Part A: Policy and Practice, 41(3), 280-285.

Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische mathematik, 1(1), 269-271.

Fielding, R. T., 2000. Architectural styles and the design of network-based software architectures. University of California, Irvine.

Fraselle J, Limbourg SL, Vidal L., 2021. Cost and Environmental Impacts of a Mixed Fleet of Vehicles. Sustainability; 13(16):9413. https://doi.org/10.3390/su13169413

Gamma, E., Helm, R., Johnson, R., Johnson, R. E., & Vlissides, J., 1995. Design patterns: elements of reusable object-oriented software. Pearson Deutschland GmbH.

Kaye D., 2003. Loosely Coupled: The Missing Pieces of Web Services. RDS Strategies LLC.; ISBN 9781881378242

Martin, J., 1983. Managing the data base environment. Prentice Hall PTR.

Montreuil, B., 2011. Toward a Physical Internet: meeting the global logistics sustainability grand challenge. Logistics Research, 3(2), 71-87.

Montreuil, B., Meller, R.D., Ballot, E., 2013. Physical Internet Foundations. In: Borangiu, T., Thomas, A., Trentesaux, D. (eds) Service Orientation in Holonic and Multi Agent Manufacturing and Robotics. Studies in Computational Intelligence, vol 472. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35852-4_10

Nuzzolo, A., and Comi, A., 2015. Urban freight transport policies in Rome: Lessons learned and the road ahead. Journal of Urbanism: International Research on Placemaking and Urban Sustainability, 8(2), 133-147.

Strulak-Wójcikiewicz, R., and Wagner, N., 2021. Exploring opportunities of using the sharing economy in sustainable urban freight transport. *Sustainable Cities and Society*, *68*, 102778.

Tawfik, C., & Limbourg, S., 2019. Scenario-based analysis for intermodal transport in the context of service network design models. Transportation research interdisciplinary perspectives, 2, 100036.

Verdonck, L., Caris, A. N., Ramaekers, K., & Janssens, G. K., 2013. Collaborative logistics from the perspective of road transportation companies. Transport Reviews, 33(6), 700-719.