Smart Mobility for All: A Global Federated Market for Mobility-as-a-Service Operators

Franco Callegati, Maurizio Gabbrielli, Saverio Giallorenzo, Andrea Melis, Marco Prandini

Smart Mobility for All

A Global Federated Market for Mobility-as-a-Service Operators

Franco Callegati[1], Maurizio Gabbrielli[1,2], Saverio Giallorenzo[1,2], Andrea Melis[1], Marco Prandini[1]

Abstract—Multi-modal travelling is a common phenomenon. However, planning multi-modal journeys is still an unstructured and time-consuming experience for customers: they lose time assembling a comprehensive plan out of disparate data, spread over a multitude of information systems — each corresponding to a different company responsible for one of the legs in the journey. Also transport operators are affected by the sparsity of the transportation market, as they might lose potential customers who could not find or know about their services. In this paper, we propose Mobility as a Service (MaaS) as a solution to such problems. Key element of MaaS is that MaaS operators can aggregate solutions of multiple providers to deliver dynamic, transparent multi-modal travels to their users, who experience transportation as managed directly by a single operator. However, given the volume and sparsity of the transportation market, we argue that MaaS operators cannot rely on one-to-one, custom contracts of usage with single mobility operators. Instead, we envision the creation of platforms that automatise the marketing of services for mobility among many mobility providers.
In this work, we detail the required features of a general software platform for such a MaaS market. In particular, we provide a precise definition of MaaS through the *MaaS Stack* — a tiered view of the components needed by entities to join the MaaS market. Then, through the lens of the MaaS Stack, we elicit the features of an enabling software platform. Finally, to validate our approach, we present a compliant prototype, called SMAll, and discuss its main design choices, among which: *i*) how SMAll supports the creation of a federation-based MaaS market and *ii*) how microservices — an emerging architectural style that fosters cohesiveness and minimality of components — enhance flexibility and let the platform and the services of its members efficiently scale according to dynamic demands.

Index Terms—Mobility as a Service, Microservices, Federated Platforms

1 Introduction

Issues of Multi-modal Travelling: Multi-modal travelling is a common phenomenon. Commuters, tourists, and travelling workers are used to compose trips out of legs covered with disparate means: bike, car (personal, rented, hailed, or shared), bus, train, boats, and planes. Usually, each "hop" requires interaction with a different operator and, mostly, with a different information system, since mobility resources are administrated and owned by a scattered plethora of mobility operators. As noted in [1], this is due to regulatory and logistic constraints that favour site-specific solutions. Hence, the experience of multi-modal travelling results often into a discontinuous flow of interaction, scattered over many applications, having a negative effect on both mobility providers and customers. The former suffer *opportunity costs* due to the loss of potential clients that could not find or know about their services. On the other side, the customers undergo many inconveniences, culminating in a sensible waste of time. Reasons comprise:

- *uneven experience*: customers may have to plan their trips over separate systems and different media with inconsistent interfaces and flow of interaction (e.g., calling a taxi via phone and then continuing the trip on train, whose ticket was booked online);
- *access issues*: although multi-modal planning services have become freely available (e.g., mapping services provided by Google and Microsoft), customers still need to find out *what* provides information on their trips. They have to look for places, phone numbers, and dedicated applications or websites to retrieve information and book the trips.
- *interaction issues*: once customers found *what* means provide information on their trips, they have to negotiate *how* to get those information, dealing with multiple authentication systems, extracting data from different representations, and aggregating them to obtain a comprehensive plan of the whole travel. Each step increases the risk of introducing inconsistencies or missing key pieces of information.

In addition to the time lost by customers and missed opportunities of mobility providers, there are strong concerns regarding *data replication and security*. When users interact with multiple, separate systems within the same travel (i.e., to plan/book one of its legs), they are likely to replicate information that is already present in another system. However, since they have to manually replicate such data (e.g., their IDs, dates of the trip, or personal needs), they could introduce discrepancies among legs. As an example, consider the steps from a multi-modal travel-planning application to the actual booking of the travel. Although the travel-planner holds information on each leg of the travel, the customer has to manually replicate a subset of such data when she books each leg, managed by a different operator. Also the security of the data of the travellers plays an important role [2]. Customers have to provide their personal data to systems that guarantee uneven security measures, making difficult or even impossible to assess the level of security of the whole process.

Mobility as a Service: A possible solution to the issues illustrated above is the creation of a unifying framework

[1] *Università di Bologna, Via Zamboni, 33, 40126 Bologna, Italy*
[2] *INRIA, France*
Manuscript submitted on 18th July 2017.

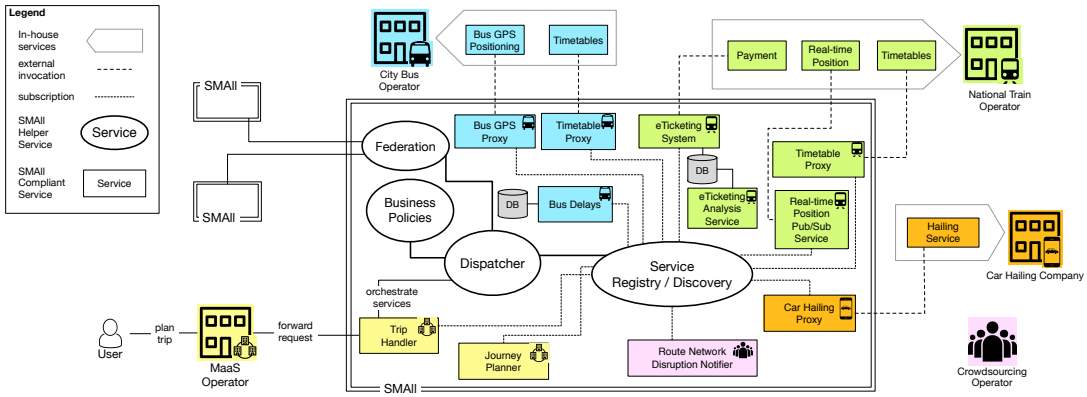

Figure 1. Representation of the SMAll architecture.

for mobility that supports the coordination of different transportation systems. Such an idea has been envisioned and described in several works in the last decade [1], [3], [4], [5]. More recently, this idea has materialised in some practical applications inspired by the concept of Mobility as a Service (MaaS) — see for example [6]. Analogously to the case of Cloud Computing (conveyor of the everything-as-a-service paradigm), MaaS hides a dynamic infrastructure of different travel agencies into a consistent interface: this makes MaaS users experience travelling as provided by a single agency. Ideally, a MaaS provider, also called MaaS operator, shall provision its users with information and procedures for discovering, planning, booking, and guiding journeys, combining any variety of means of transportation. To the final user, the provisioning of *mobility resources* (i.e., information on transportation and the actual transits) is transparent wrt the actual provider of the service. Since mobility resources are administrated and owned by disparate mobility providers, we argue that the leading economic model of MaaS markets is that of federations of providers, each trading its mobility resources. In such a federated market, MaaS operators dynamically partner with each other whilst preserving their individual autonomy and without a centralised regulation authority — which, in the case of transportation, would be practically impossible to appoint.

Contributions: In this paper we present two main contributions. First, we introduce the *MaaS Stack* (§ 3): this is the first tiered view that provides a structure for the, so far, informal concept of MaaS. The MaaS Stack originates from our discussions with companies interested in entering the market of Mobility as a Service, as well as from investigations conducted within the EU EIT Digital project SMAll[1]. We deem our view useful to isolate and clarify which elements must be in place for mobility operators to join the MaaS market, possibly becoming themselves MaaS operators. The second contribution is the presentation of a platform called *Smart Mobility for All* (SMAll)[2,3] (§ 4), that we are currently developing to support the creation of a federation-based MaaS market and which is structured according to the principles of the MaaS Stack. SMAll facilitates the publication, automatic retrieval, and orchestration of functionalities for mobility, provided by different mobility operators. The platform builds on the concept of Federated Cloud Computing [7] and maintains an open approach wrt the possible members of the federation: MaaS operators, traditional transport agencies, and other players that trade information linked to mobility, like weather forecasts or crowd-sourcing communities [8].

Given its open approach, SMAll can host and enhance any service for mobility already present in the market. As an example, we consider an actual pilot developed as part of the mentioned EU project. In the pilot, a mobility operator had a customised multi-modal journey planner (Open Trip Planner). The operator deployed the planner in SMAll and refined its outputs with real-time GPS data on public transport vehicles, provided by other operators present in SMAll. More in general, the provider of the planner can automatise the retrieval of real-time data offered by any other mobility operator in the market. Note that, given the dynamic nature of contracts of usage in SMAll, real-time data is accessed on-demand and its retrieval is limited/optimised to the actual queries received from the users of the planner. In § 2 and § 5 we provide further examples of how SMAll can host and/or expedite the enhancement of services already present in the transportation market.

Concluding our introduction, we highlight that SMAll is developed following the microservices paradigm [9] and members are strongly supported in publishing their functionalities as microservices. As remarked in § 4.1, such architectural choice positively impacts on both SMAll and the resources deployed by members, which enjoy great flexibility, gradual deployment and continuous integration, eased software maintenance and, most important, efficient scalability according to dynamic demands.

2 Overview

In this section, we overview the concept of MaaS and illustrate the main features of SMAll with a representative instantiation of our platform, depicted in Figure 1. In the remainder of the paper, we use the term *service* to indicate an application deployed within or outside SMAll, while we use the term *microservice* to indicate an instance of a service

1. Project description: `https://goo.gl/WKnnSW`
2. Wiki: `https://github.com/small-dev/SMAll.Wiki/wiki`
3. Deployable platform: `https://hub.docker.com/u/smallproject/`

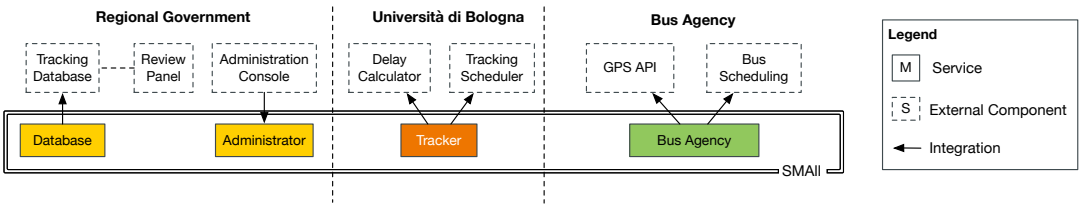

Figure 2. Representation of the BusCheck Pilot.

within SMAll[4]. In Figure 1, the coloured entities outside of the boundaries of SMAll (bordered with double lines) are providers of mobility resources and MaaS operators. These are public transportation agencies, private companies, and online communities. By entering the platform, each member can be dynamically federated with the other agents already present. Once in SMAll, members can deploy their own functionalities as microservices, e.g., in Figure 1, the City Bus operator deploys three services: the first two are Bus GPS Proxy and Timetable Proxy, which function as wrappers for some pre-existing applications deployed in-house by the operator. The third service, Bus Delays, is completely contained in the platform and orchestrates the other two services of the Bus Operator to calculate the delays of buses by comparing the actual GPS position of the rides with the expected scheduling from the timetables.

SMAll provides helpers (e.g., Registry / Discovery and Dispatcher) to publish, discover, compose, and regulate the usage of the deployed microservices. Considering the example above, although all the microservices belong to the Bus Operator, all the invocations from the Bus Delays service are routed and managed by the Dispatcher. The Dispatcher also enforces the usage Business Policies defined by the owner of the invoked service, e.g., it can refuse to proxy the invocation to the addressee as well as to delay frequent requests if they exceed the rates established by the owner.

Business Policies are fundamental for marketing on-demand services. As an example, consider the case in which the City Bus Operator integrates the crowd-sourced data on Route Network Disruption in its Bus Delays service (e.g., to forecast day-long delays on the interested routes). With SMAll, using such crowd-sourced data does not require the presence of pre-existing contracts of usage between the Bus Operator and the Crowdsourcing Operator. By accepting the business policies formalised by the Crowdsourcing Operator, the Bus Operator can dynamically access (and pay for) the Disruption information.

Finally, different SMAll installations can be federated as well, so that region-wide instances can constitute a federation of international- and world-wide platforms. Consider, for example, a MaaS operator that wants to provide transportation solutions to its users travelling abroad. It would be unthinkable for the MaaS operator to foresee and stipulate contracts of usage in advance with all the possible foreign transport agencies. On the contrary, with SMAll a MaaS operator can automatise the dynamic aggregation of foreign federated services for its users, letting them access transport solutions of other operators. Moreover, like the other members of the platform, also MaaS operators can deploy services. For example, in Figure 1 the MaaS Operator deploys a Journey Planner and a Trip Handler. The latter, in particular, is the service that orchestrates the dynamic multi-modal trips for the users of the MaaS Operator. To do that, Trip Handler interacts with the Dispatcher to reach and orchestrate the other federated services: it uses information on scheduling, availability, disruptions, and the position of buses, trains, and on-demand cars to dynamically plan a multi-modal trip, booking and paying the rides for the user.

A Motivating, Real-World Example: We now illustrate the proposed approach with a real-world use case developed within the recent EU EIT Digital project SMAll.As part of the project, we investigated the suitability — in terms of development, interoperability, and scalability — of SMAll wrt the creation of new smart mobility applications, possibly integrating pre-existing services.

As a real-world example, we report one of such applications, which we implemented as a pilot, called BusCheck. The pilot, commissioned by the Department of Transportation of the government of the Emilia-Romagna (ER) region (Italy), aims at recoding and displaying the quality of service of the buses in the Bologna province. Figure 2 represents the architectural view of our solution, composed of interacting services (continuous boxes) and external functionalities (dotted boxes) owned and provided by three organisations: the ER regional government, the University of Bologna, and TPER, the bus agency of the Bologna area. In Figure 2, we omit Registry, Discovery, and Dispatcher helpers that enable interaction among the deployed services (explained in § 4.1).

As shown in Figure 2, BusCheck emerges from the composition of four services deployed within SMAll (double-line in Figure 2): *i*) *Administrator* is a service owned by the ER government and used by operators to schedule and issue the tracking of buses. Operators interact with the service through a in-house client GUI outside SMAll. *ii*) *Tracker* is a service developed and maintained by the University of Bologna. It implements the actual logic to track buses. The service relies on two sub-services: a Tracking Scheduler that, based on the timetables of the tracked bus, triggers the retrieval of its real-time position; a Delay Calculator that computes the divergence between the expected and actual positions of the tracked bus. *iii*) *BusAgency* is a service maintained by the Bus Agency. It exposes to the Tracker the static and real-time data on all the vehicles of the bus agency. *iv*) *Database* is a service owned by the ER government that interacts with the Tracker, receiving data on delays and serving them internally to regional operators for both real-time and static inspection.

4. Hence, to a service correspond one or more copies of the same microservice that implements its functionalities.

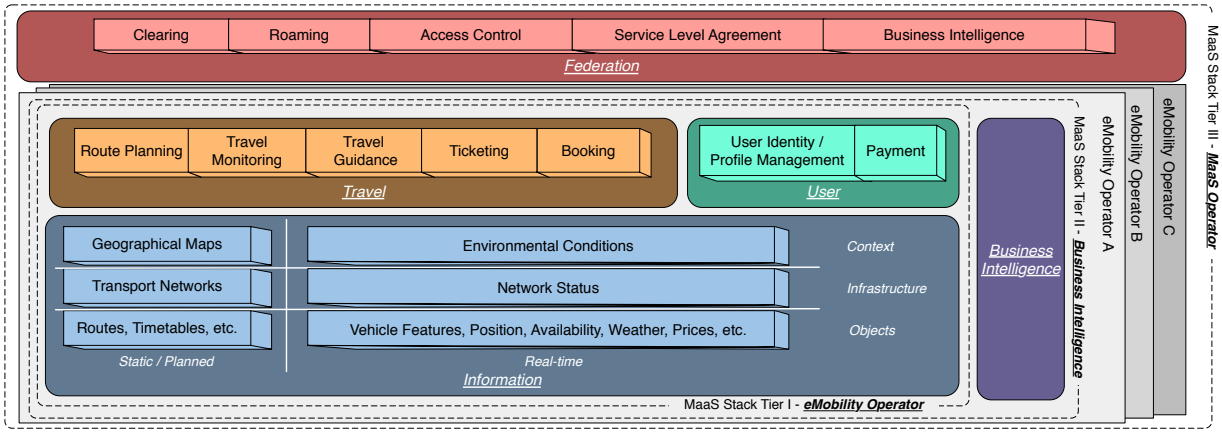

Figure 3. The MaaS Stack

3 The MaaS Stack

For the clarification of the concept of MaaS, we deem useful to define it as a tiered structure, called the MaaS Stack. We use such partition in § 4 to analyse the elements of MaaS markets and the solutions we integrated in SMAll.

Figure 3 represents the MaaS Stack, comprising 3 tiers (dashed lines): *eMobility Operator*, *Business Intelligence*, and *MaaS Operator*. Inside the tiers, we identify 4 macro-layers of services (rounded rectangles), each building on top of the others: the *Information* layer contains basic services like timetables and real-time positions; *Travel* and *User* services build on Information ones to offer more advanced features to users; *Business Intelligence* services analyse data on performance and usages of the aforementioned services, providing insight to their owners; *Federation* services let operators trade their solutions and form dynamic partnerships.

In the next sections we analyse in depth the tiers of the MaaS Stack and the layers that characterise them.

3.1 Towards Mobility as a Service

Recently, traditional transport agencies have been publishing the data of their transportation solutions. Airline companies [10], train operators [11], and city-to-region-wide bus operators have been compelled to open the data regarding their transport systems to integrate with (de-facto) standards [12]. Beside traditional transport agencies, new companies entered the transportation market offering functionalities for mobility such as mapping, travel guidance, multi-modal journey planning and booking. The latests novelty on the transportation market scene are hailing companies like Uber and Lyft: "virtual" agencies that offer software functionalities to enable transport solutions. This scenario characterises the first tier of the MaaS Stack where isolated entities, called *eMobility operators*, provide functionalities for mobility called *eMobility services*..

3.2 MaaS Stack Tier I | eMobility Operators

Definition 1 (eMobility Service). A software functionality for mobility, provided in a machine-readable form.

For example, an eMobility service could give access to airline/train/bus schedules in machine-readable formats [12], [13], [14]. Support for machine-readable formats is key to enable the dynamic composition of services, so that information can be automatically processed, enriched, aggregated, etc., and exposed to other services in the same fashion. Figure 3 shows the layered taxonomy of eMobility services in the MaaS Stack. The mobility-specific services fall into two macro-categories: *Information* and *Travel* ones; then, there are what we define *User* services, such as user identity management, user preferences, payment circuits, etc., which are not specific to mobility. A listing of the fundamental services in these categories includes the following.

Information services allow users to access basic data needed for transportation purposes.

- *Static / Planned* data is stable or seldom updated, regarding elements like maps, infrastructures (e.g., bus stops, parking, rails, docks), and timetables of transport services.
- *Real-time* data report the status of the system. This ranges from unexpected infrastructural unavailability to current weather conditions, GPS position of vehicles, available seats/spots, traffic, delays, strikes, etc..

As depicted in Figure 3, Information services can be stacked as well. We illustrate the concept considering the *Static / Planned* information stack: the basic data on vehicles regarding e.g., *Routes* and *Timetables* provide a base for the static *Transport Network*. This holds also for *Geographical Maps* with respect to the transport network.

User services provide functionalities loosely related to transportation but necessary to interact with users.

- *User Identity / Profile Management* services allow to authenticate and authorise users, and also include functions to manage their preferences and historical records.
- *Payment* services handle transactions to pay transportation solutions and (possibly) the access to eMobility services.

Travel services mainly build upon Information ones.

- *Journey Planning* services find journeys, possibly multimodal, between two points. They work on static data, possibly integrating real-time one for dynamic results.
- *Travel Guidance* assist the user with real-time travelling information wrt her position (e.g., turn-by-turn guidance).
- *Travel Monitoring* services track the position of the user and notify other subscribed applications of check-ins/-outs and other events related to her movements.

Booking services use Information and Payment services to place a reservation for the user (e.g., seats, spots, etc.).

Ticketing integrates Booking services to create and deliver transportation tickets (i.e., access tokens) to users.

In the first tier of the MaaS Stack, we consider only single eMobility operators (i.e., operators that do not use and integrate the services of other operators).

Definition 2 (eMobility Operator). An entity that owns, administrates, and exposes eMobility services.

Intuitively, an eMobility operator is any entity (company, association, etc.) that publishes and orchestrates a set of eMobility services directly administrated by itself. An example of tier I eMobility operator is the National Train Operator in Figure 1: it exposes services to buy tickets online and to publish timetables and the real-time position of vehicles in machine-readable standards. Our definition of eMobility operators comprises also providers of services not directly linked to transportation solutions, for example weather forecasts: indeed, they provide an important information on mobility and can enter the MaaS market as well.

3.3 MaaS Stack Tier II | Business Intelligence

The second tier of the MaaS Stack still focuses on single eMobility operators but it enriches the taxonomy of eMobility services with the category of *Business Intelligence* [15]. This category of services is separated from first-tier ones for two reasons: first they are not meant for users but rather for eMobility operators, and second, they span over all first-tier services by monitoring and analysing their usages. The aim of Business Intelligence services is to provide insight on the performances of eMobility services.

In Figure 1, an example of second-tier Business Intelligence service is the eTicketing Analysis Service. The service can access the data of the eTicketing System and, e.g., can suggest new pricing policies to the National Train Operator as well as reporting rarely used routes that could be merged/discarded. More generally, other examples of Business Intelligence services comprise reporting on the usage of the published eMobility services, analysis of the quality of transit systems (e.g., relating the discrepancies between scheduled trips and real-time delays), monitoring the profitability, sustainability, and reliability of the provided services and determining trends and making predictions on future usage, for capacity planning and policy definition.

3.4 MaaS Stack Tier III | MaaS Operators

As mentioned in § 1, the market of eMobilty operators is a very scattered one. The concept of Mobility as a Service originates from the economic opportunity of bridging the gaps between operators, both cutting down overhead for users and enabling synergistic strategies among transport providers. For the creation and success of such a MaaS market, it is imperative that eMobility operators can trade and use said services on-demand. Such high degree of flexibility (and trust) is typical of *federations* [7].

Definition 3 (MaaS Operator). An eMobility operator federated with other eMobility operators. A MaaS operator provides to its users eMobility and transit services of other operators as its own. The usage of such foreign services undergoes formal business policies.

In the context of MaaS, when eMobility operators federate, they accept to adopt common technologies and formal *business policies*. Such technology standards and regulations are critical for a marketplace where eMobility and transportation services are traded like stocks, i.e., dynamically (not regulated by long-term, static contracts) and on-demand. Federated operators establish business policies to mechanise the trading of their services. The aim is to let users integrate eMobility services and transportation solutions of "foreign" operators into their travelling experience. The principle, already envisioned in [1], resembles that of *roaming* of GSM phone networks [16], where users connect through the services of another phone company when travelling outside the geographical coverage area of the home network.

Definition 4 (MaaS Roaming). Users of a MaaS operator can transparently use eMobility and transit services of other, federated operators.

As an example, consider the MaaS Operator in Figure 1. Being federated with the National Train Operator and The City Bus Operator, it can offer multi-modal journeys that span different means of transportation (rail and road) and have wide-to-narrow scopes (inter-city and intra-city). For example, the MaaS Operator can leverage the available business policies so that its users can plan and purchase a trip (through its journey planner) associated with an eTicket bought from the National Train Operator which also comprises 5 trips of the City Bus Operator. The synergy benefits all partners: the MaaS Operator provides (and it is paid for) a comprehensive service to its users; the National Train Operator acquires users and can charge for the access to its eTicketing system; the City Bus Operator acquires new users that (probably) would otherwise have taken a taxi due to the overhead of looking for the right route and where to buy the needed tickets. Our example introduces the last fundamental element of the third tier of the MaaS Stack: *Clearing* services, i.e., eMobility services that account for roaming usages and compensate operators according to the established business policies. Callegati et al. analyse in [17] a similar scenario, where transport companies within the same region share a unique ticket, requiring a clearing system to manage the redistribution of the profits according to the policy agreement of ticket sold and validated.

4 The SMAll Architecture

4.1 A Market of Microservices

Following the lesson of Cloud Computing [18] (and the related SaaS/PaaS/IaaS stack), we argue that MaaS providers will require tools and infrastructures to harness the heterogeneous landscape of eMobility operators. We choose microservices [9] as the enabling technology for an on-demand marketplace after the observation that operators (e.g., the Bus Operator in Figure 1) already have a collection of legacy software systems that address some specific issues (e.g., the Bus GPS Positioning and the Timetables services) and that other operators (e.g., the National Train Operator in Figure 1) are willing to pay to access them. In practice, operators would like to include specific external functionalities (e.g., a bus

tracking service) in their own services rather than use (and pay for) a bundle of unneeded functionalities (e.g., a real-time planner that includes the mentioned bus tracking capability). Microservice architectures achieve such degree of granularity. This feature recently drove companies like Amazon and Netflix to adopt microservice architectures, enabling them to harness the complexity of their large software base. Indeed, microservice architectures bring fundamental features for on-demand provisioning, among which: independent development cycles, per-usage resource allocation (limiting the allocation for unneeded bundled functionalities), and freedom to use task-specific technologies.

As expected, microservices come with some trade-off: loosely-coupled microservices communicate via message passing, which requires proper routing and may suffer latencies and failures; many requests can overload a service, hence it should prevent outages by limiting them and/or scaling accordingly; microservices are heterogeneous but their data-formats and Application Programming Interfaces (APIs) should be homogeneous to foster compositionality.

For these reasons, SMAll strives for standardization of data-formats and APIs. Moreover, it provides infrastructural tools to cope with most of the aforementioned issues: orchestration abstractions to streamline the composition of available services (via the Jolie programming language [19], [20]); data-format conversion functionalities; service registries and dispatchers to both store the definition and address of all the services deployed on the platform and to route requests to them; business intelligence outlets for auditing and performance indicators on the usage of services. In addition, we investigated technologies to counteract security issues [2], to enforce business policies for the dynamic access to services, to federate different deployments of SMAll, and to regulate the usage of services among members.

We now proceed to address the main issues of MaaS federated markets, describing the elements of SMAll that deal with them. In doing so, we follow the MaaS Stack from the bottom up. We first focus the needs of single eMobility operators within the first two tiers, broadening our view to MaaS operators in the third tier.

4.2 Tier I and II

Whilst tier I and II are stacked and their respective services differ from a user perspective (travellers for the first tier, operators for the second one), at the architectural level they share the same needs and components. Hence, in this section, we consider them together.

Concerning these two tiers, we fixed some basic requirements in the design of SMAll that we deem necessary within a platform for single-tenant microservice deployment:

- sandboxing [21] for development and security;
- scaling both horizontally (i.e., create and remove copies of the same service) and vertically (i.e., increase and decrease the resources available to a microservice);
- publication and discovery of services and the related Application Programming Interfaces (APIs);
- orchestration of services.

Below, we detail the mentioned requirements and discuss the elements of the SMAll platform that address them.

Deployment — Sandboxing and Scaling: virtualisation is the standard solution for cloud-based deployment of services [18]. The administrator of a service creates an isolated virtual machine, i.e., sandboxed, wrt the others and deploys her service on it. Then, single virtual machines can be scaled vertically and, by managing the number of copies of the same machine, horizontally. However, virtual machines have several shortcomings: they entail important costs due to the need for dedicated resources, these resources could be wasted in idle cycles, and preparing and deploying full virtual images takes sensible time. Hence, in SMAll we chose *containerisation* [22] as suitable solution that balances costs, time, and ease of development with a flexible and secure deployment. On these regards, SMAll can integrate techniques to optimise the deployment of microservices based on a description of the target configuration [23].

Beside deployment, microservices are mainly involved in orchestration, to which we dedicate the next two paragraphs. In the first, we describe why and how services should be indexed into a registry for discovery. In the second, we show how to support the orchestration of discovered services.

Orchestration — Registry and Discovery: in cloud-based platforms like SMAll, the address of a microservice is dynamically determined at the time of the deployment and can even change during the life-cycle of the microservice (e.g., due to migrations). For this reason, SMAll does not provide the direct address of a deployed microservice. Instead, following a pattern called registry-and-discovery — adopted by other cloud platforms [24], [25], [26], [27], [28] — when a programmer deploys a service in SMAll, she also registers it, with its description and APIs, into a Registry. As reply, the Registry returns a unique identifier of the microservice, which works as its reference address.

A registered service becomes visible to (allowed) users through the Discovery service, i.e., a SMAll helper dedicated to query the Registry to find services that match the requirements expressed by a user. Indeed, SMAll supports users and access policies to let the owner of a service define which member can discover (and interact with) it. This is useful also within the borders of a single eMobility operator, where different departments deploy services handling confidential information, to avoid dangerous leakages [2].

Finally, regarding API definition, in SMAll we chose to support RESTful [29] and Jolie ones. On the one hand, we chose to support RESTful interfaces for compatibility, given the current adoption of RESTful technologies. On the other hand, we argue Jolie interfaces to be more flexible than RESTful ones (e.g., they are not constrained within HTTP verbs). Moreover, Jolie interfaces enjoy desirable features like out-bound and in-bound checking for compatibility wrt the specified API [30]. Hence, when present, we preserve Jolie interfaces of microservices and provide tools[5] for the automatic conversion to RESTful ones.

Orchestration — Routing: The last feature we consider here is routing of requests to registered services, which in SMAll is embodied by the Dispatcher service. As an example of interaction with the Dispatcher, consider Figure 4

5. The Jolie REST router: `https://github.com/jolie/jester`

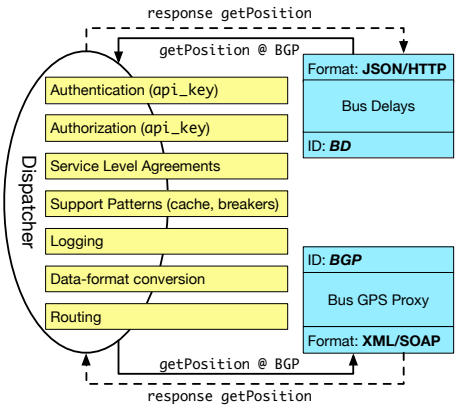

Figure 4. Example workflow of the SMAll Dispatcher.

that depicts the invocation of the Bus GPS Proxy from the Bus Delays service (cf. Figure 1). In Bus Delays the programmer writes the orchestration code, labelling the invocations to the Bus GPS Proxy with its identifier `BGP` — as mentioned, programmers can obtain microservice identifiers at deployment-time or through the Discovery. At runtime, each invocation done by the Bus Delays service passes through the Dispatcher, which interprets the label assigned to the invocation and redirects it to the actual deployment address of the Bus GPS Proxy. The Dispatcher handles the routing of the response back to the invoker.

The Dispatcher also plays a central role in the development of microservices in SMAll (similar to API Gateways [26], [31]) and supports the advanced features, like access policies and service level agreements, of the third tier of the MaaS Stack. These features (rectangles in Figure 4) comprise:

- *Authentication* and *Authorisation*, forwarding only requests allowed to interact with the invoked service;
- *Service Level Agreements* which e.g., regulate the rate of calls per time unit and certify the respect of availability and responsiveness contracts;
- *Support Design Patterns* for microservices, like circuit breakers [32], caches, etc., which normally would require a direct integration within (and modification of) the microservices, as done e.g., with Netflix Hystrix [33];
- *Logging*, which is useful for debugging and security;
- *Data and Channel conversion* for the seamless integration of heterogeneous services, e.g., in Figure 4 the request of the Bus Delays service uses HTTP and JSON while the Bus GPS Proxy uses XML over SOAP.

4.3 Tier III

The third tier contains the most advanced features of SMAll, devoted to the creation of a global MaaS market.

We note that SMAll encompasses two types of federations. The first one is at the level of eMobility operators, which can trade and access their services (becoming MaaS operators). These federations are defined *dynamic* because *i*) they exist at runtime, according to the automatic enforcement of the Access Policies and Service Level Agreements of the invoked service and *ii*) they live as long as their related transaction between the parties. The second type of federation concerns SMAll instances. Here, the federation is *static*, i.e., the owners of SMAll instances define agreements regarding the inter-communication technologies [34], the security, reliability, and availability of their link, and the security requirements within their platforms. With federated SMAll instances, eMobility operators belonging to distinct instances can trade their services and establish dynamic federations that span different geographic contexts.

The fundamental components that allow SMAll to form a unique market of eMobility services are the Discovery and the Dispatcher. Indeed, once these two components are aware of the presence of other instances of SMAll, they are able to automatically route discovery queries and service invocations towards the other federated platforms. To do this, both components forward their (respective discovery and invocation) requests towards their equivalent in the targeted platform. The forwarded request is handled as coming within the targeted platform. This is the context where the advanced features of the Dispatcher heavily come into play to enforce Access Control and Service Level Agreements.

Finally, dynamic federations of eMobility operators enable the support for roaming, the hallmark of Mobility as a Service, i.e., that MaaS users can integrate into their travelling experience the eMobility services and the transportation solutions of other operators. Clearing services are the last piece that completes the picture in SMAll, as they compensate usages of transport solutions as well as of eMobility services, according to the contract agreements.

5 Related Work and Conclusion

In this paper, we argued how Mobility as a Service (MaaS) represents a feasible solution to the problems of multi-modal travelling. On a broader perspective, we illustrated how MaaS concretise a steady trend of research on transportation systems [1], [3], [4], [5] that foresees disparate and sparse transportation networks unified within communicating frameworks for mobility. In doing so, we informally introduced the characteristics of MaaS operators and how they shall facilitate the dynamic provisioning of multi-modal transportation to their users. Then, we formalised MaaS through a novel, structured view, called the MaaS Stack. Finally, we presented SMAll, our prototype platform for the creation of a federation-based global market for MaaS.

Regarding related work, to the best of our knowledge, both the MaaS Stack and SMAll have no direct work to compare with. Indeed, the MaaS Stack is the first formal treatment on the features of MaaS markets. Similarly, SMAll is the first platform that enables the automatic creation of services for MaaS.

For the sake of completeness, we consider some platforms and applications for mobility already present in the market and discuss how they compare with SMAll. The closest platforms to SMAll are MyCicero [35] and FluidTime [36], however they adopt a closed approach where customers collaborate on a one-to-one basis and the offering and integration of services for mobility is not done by customers but handled directly by the providers of the platform. Broadening our scope, there are applications that interact with multiple mobility companies like Swiftly [37], for the analysis and management of the real-time data on traffic,

and Hannovermobil [38] for booking and ticketing of trips. Also in these cases, both applications create a one-to-many relation with the providers of data and services, which cannot directly interact with each other. Note that all the mentioned platforms and applications could be deployed in SMAll and leverage its features to automatise the inclusion of data and services provided by the federated members. A similar comparison can be drawn with applications like Bridj, RideCell, and ZipCar. These are focussed on the provisioning of proprietary vehicles to move people and goods. Once deployed in SMAll, these applications could both *i*) federate, selling to each other the access to their vehicles/customers, and *ii*) automatise the inclusion of local providers and communities [39], generating a liquid market around the same core business. Analogously, applications for multi-modal journey planning like Rome2rio, Google Transit, Hyperdia, and NaviTime, once deployed in SMAll, could automatically enrich their results with real-time (possibly crowd sourced [40]) data, for dynamic trip planning [41], also considering traffic monitoring and accident detection [42], and the collection of crowd-sourced data on cognitive distraction [43], drowsiness [44], and behaviour of drivers [45] to avoid incidents.

Although SMAll is currently at a prototypical stage, we are validating the platform with our industrial partners. As future work, we plan to provide tools to support programmers and system owners in the creation, verification, and maintenance of microservices. In these respects, cutting edge technologies for service composition like Choreographic Programming [46], [47] can help in establishing partnerships among members of SMAll (i.e., formal contracts) which can even be dynamically updated after deployment [48].

Finally, the third tier of the MaaS Stack requires technologies for the federation of SMAll instances [7] (i.e., federation of cloud clusters), the definition and the enforcement of Access Policies and Service Level Agreements [49], as well as best practices, standards, and techniques to guarantee high levels of security within the platform.

REFERENCES

[1] J. M. Parker, "Applying a system of systems approach for improved transportation," *SAPIENS*, no. 3.2, 2010.

[2] F. Callegati, S. Giallorenzo, A. Melis, and M. Prandini, "Insider threats in emerging mobility-as-a-service scenarios," in *HICSS*, AIS Electronic Library (AISeL), 2017.

[3] M. W. Maier, "Architecting principles for systems-of-systems," in *INCOSE*, vol. 6, pp. 565–573, Wiley Online Library, 1996.

[4] M. W. Maier, "Research challenges for systems-of-systems," in *2005 IEEE SMC*, vol. 4, pp. 3149–3154, IEEE, 2005.

[5] D. Giuli, F. Paganelli, S. Cuomo, and P. Cianchi, "Toward a cooperative approach for continuous innovation of mobility information services," *IEEE Systems Journal*, vol. 7, no. 4, pp. 669–680, 2013.

[6] S. Pippuri *et al.*, "Maas finland." `http://maas.fi`.

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and architectures for parallel processing*, pp. 13–31, Springer, 2010.

[8] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[9] N. Dragoni *et al.*, "Microservices: yesterday, today, and tomorrow," in *PAUSE*, Springer, 2017. to appear.

[10] S. Morrison and C. Winston, *The evolution of the airline industry*. Brookings Institution Press, 1995.

[11] A. Nash, D. Huerlimann, J. Schütte, and V. P. Krauss, "Railml–a standard data interface for railroad applications," *Computers in Railways IX, WIT Press, Southampton*, pp. 233–240, 2004.

[12] Google, "Google transit feed specification." `https://developers.google.com/transit/`.

[13] Google, "Google transit feed specification | realtime transit." `https://developers.google.com/transit/gtfs-realtime/`.

[14] CEN, "Service interface for real time information." `http://user47094.vs.easily.co.uk/siri/`.

[15] S. Negash, "Business intelligence," 2004.

[16] M. Mouly, M.-B. Pautet, and T. Foreword By-Haug, *The GSM system for mobile communications*. Telecom publishing, 1992.

[17] F. Callegati, A. Campi, A. Melis, M. Prandini, and B. Zevenbergen, "Privacy-preserving design of data processing systems in the public transport context," *PAJAIS*, vol. 7, no. 4, 2015.

[18] R. Buyya *et al.*, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *FGCS*, vol. 25, no. 6, pp. 599 – 616, 2009.

[19] Jolie Team, "Jolie programming lang.." `http://jolie-lang.org`.

[20] F. Montesi, C. Guidi, and G. Zavattaro, "Service-oriented programming with jolie," in *WSF*, pp. 81–107, Springer, 2014.

[21] V. Prevelakis and D. Spinellis, "Sandboxing applications.," in *USENIX Annual Tech. Conf., FREENIX Track*, pp. 119–126, 2001.

[22] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, Mar. 2014.

[23] M. Gabbrielli, S. Giallorenzo, C. Guidi, J. Mauro, and F. Montesi, "Self-reconfiguring microservices," in *Theory and Practice of Formal Methods*, pp. 194–210, Springer, 2016.

[24] Netflix, "Eureka." `https://github.com/Netflix/eureka`.

[25] Netflix, "Ribbon." `https://github.com/Netflix/ribbon`.

[26] Amazon, "API Gateway." `https://aws.amazon.com/api-gateway/`.

[27] Apache, "Zookeeper." `https://zookeeper.apache.org/`.

[28] WSO2, "Wso2 api manager." `http://wso2.com/api-management/`.

[29] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. PhD thesis, UC Irvine, 2000.

[30] F. Montesi, "Process-aware web programming with jolie," *Science of Computer Programming*, vol. 130, pp. 69–96, 2016.

[31] Netflix, "Zuul." `https://github.com/Netflix/zuul`.

[32] F. Montesi and J. Weber, "Circuit breakers, discovery, and api gateways in microservices," *CoRR*, vol. abs/1609.05830, 2016.

[33] Netflix, "Hystrix." `https://github.com/Netflix/hystrix`.

[34] K. Indrasiri, "Microservices in pratice - key architectural concepts of an msa," *Wso2 White Paper*, 2016.

[35] Mycicero, "http://www.mycicero.it/." `http://www.mycicero.it/`.

[36] Fluidtime, "Fluidtime." `https://www.fluidtime.com`.

[37] Swiftly, "https://goswift.ly/." `https://goswift.ly/`.

[38] Hannovermobil, "Hannovermobil." `http://www.gvh.de/service/rad-auto-carsharing/hannovermobil`.

[39] S. Mirri, C. Prandi, P. Salomoni, F. Callegati, A. Melis, and M. Prandini, "A service-oriented approach to crowdsensing for accessible smart mobility scenarios," *MIS*, vol. 2016, pp. 1–14, 2016.

[40] A. Melis, S. Mirri, C. Prandi, M. Prandini, and P. Salomoni, "A microservice-based architecture for the development of accessible, crowdsensing-based mobility platforms," in *CTS*, pp. 498–505, IEEE, 2016.

[41] J.-Q. Li, K. Zhou, L. Zhang, *et al.*, "A multimodal trip planning system incorporating the park-and-ride mode and real-time traffic/transit information," in *ITSC*, vol. 25, pp. 65–76, 2010.

[42] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE TITS*, vol. 1, pp. 108–118, Jun 2000.

[43] M. Miyaji, H. Kawanaka, and K. Oguri, "Driver's cognitive distraction detection using physiological features by the adaboost," in *ITSC*, pp. 1–6, IEEE, 2009.

[44] K. Hayashi, K. Ishihara, H. Hashimoto, and K. Oguri, "Individualized drowsiness detection during driving by pulse wave analysis with neural network," in *ITSC*, pp. 901–906, IEEE, 2005.

[45] M. Miyaji, M. Danno, and K. Oguri, "Analysis of driver behavior based on traffic incidents for driver monitor systems," in *IVS*, pp. 930–935, IEEE, 2008.

[46] M. Carbone and F. Montesi, "Deadlock-freedom-by-design: multiparty asynchronous global programming," in *POPL*, pp. 263–274, 2013.

[47] S. Giallorenzo, *Real-World Choreographies*. PhD thesis, Università degli studi di Bologna, 2016.

[48] M. Dalla Preda, S. Giallorenzo, I. Lanese, J. Mauro, and M. Gabbrielli, "AIOCJ: A choreographic framework for safe adaptive distributed applications," in *SLE*, pp. 161–170, Springer, 2014.

[49] P. Patel *et al.*, "Service level agreement in cloud computing," 2009.