ROUTE: A Framework for Customizable Smart Mobility Planners

Fahed Alkhabbas*, Martina De Sanctis†, Antonio Bucchiarone ‡, Antonio Cicchetti §, Romina Spalazzese*,
Paul Davidsson*, and Ludovico Iovino †
*Internet of Things and People Research Center, Malmö University, Sweden
{fahed.alkhabbas, romina.spalazzese, paul.davidsson}@mau.se
†Gran Sasso Science Institute, Computer Science Department, L'Aquila, Italy
{martina.desanctis,ludovico.iovino}@gssi.it
‡Fondazione Bruno Kessler, Trento, Italy
bucchiarone@fbk.eu
§IDT Department, Mälardalen University, Västerås, Sweden
antonio.cicchetti@mdh.se

***Abstract*—Multimodal journey planners are used worldwide to support travelers in planning and executing their journeys. Generated travel plans usually involve local mobility service providers, consider some travelers' preferences, and provide travelers information about the routes' current status and expected delays. However, those planners cannot fully consider the special situations of individual cities when providing travel planning services. Specifically, authorities of different cities might define customizable regulations or constraints of movements in the cities (e.g., due to construction works or pandemics). Moreover, with the transformation of traditional cities into smart cities, travel planners could leverage advanced monitoring features. Finally, most planners do not consider relevant information impacting travel plans, for instance, information that might be provided by travelers (e.g., a crowded square) or by mobility service providers (e.g., changing the timetable of a bus). To address the aforementioned shortcomings, in this paper, we propose ROUTE, a framework for customizable smart mobility planners that better serve the needs of travelers, local authorities, and mobility service providers in the dynamic ecosystem of smart cities. ROUTE is composed of an architecture, a process, and a prototype developed to validate the feasibility of the framework. Experiments' results show that the framework scales well in both centralized and distributed deployment settings.**

***Index Terms*—Multimodal Journey Planners, Software Framework, Multi-tier Architecture, Smart Mobility**

I. Introduction

Multimodal journey planners have been used worldwide for decades. They support travelers in planning and executing their journeys, considering some travelers' preferences. Nowadays, we are witnessing the rapid transformation of our traditional cities into smart cities. Realizing a smart city ecosystem requires involving a multitude of actors and enabling a wide spectrum of novel services in various domains, including smart mobility [1]. Smart cities have their own technical infrastructures and needs and might obey different regulations. For instance, the cities' authorities can define different constraints on the number of people who can gather in a place, e.g., due to a pandemic. Moreover, the authorities can define mobility constraints in cities for different reasons, including construction works. Furthermore, mobility service providers can adapt the paths and timetables of their travel media, e.g., due to the defined constraints. However, the majority of the existing planners do not consider such individual needs of the cities when providing planning services.

Moreover, the fast adoption of the Internet of Things (IoT) and Artificial Intelligence technologies in smart cities could enable the existing planners to evolve and provide novel services. Specifically, the planners could leverage advanced monitoring features, providing information to improve the provided plans' quality. For example, by exploiting the IoT network in a smart city, a planner could recommend travelers take the least noisy routes with good air quality. Additionally, travelers could provide useful information about the status of their trips. By exploiting the data collected by IoT things or from travelers, planners could generate new trips or adapt the ongoing ones to cope with dynamic changes in the smart cities for the convenience and engagement of the users.

A few studies have investigated smart mobility planning from different perspectives, including user engagement [2], [3], integrating IoT and/or AI technologies [4]–[6], and self-adaptation [7]–[9]. However, no work aims at developing the new generation of travel planners, considering all those perspectives. Specifically, little work was done from the software engineering perspective with a focus on software architectures. Towards bridging this gap aiming at overcoming the shortcomings of the existing planners, in this paper, we propose *ROUTE*, a fRamework fOr cUsTomizable plannErs. *ROUTE* is composed of an architecture, a process, and a first prototype[1] that we developed to validate its feasibility and evaluate its performance and scalability. The results of the experiments show that the framework scales well in both centralized and distributed deployment settings. *ROUTE* supports a wider set of actors, such as authorities, mobility service providers besides travelers, compared to traditional planning systems that mainly focus on travelers. Consequently, *ROUTE* better supports communities in smart cities ecosystems. For instance,

[1] The prototype of ROUTE is available at http://139.177.202.145:8080/

ROUTE enables authorities to define constraints on movements to manage the current pandemic and thus slow down the spread of the virus and speed up the recovery from the pandemic.

The remainder of this paper is organized as follows. Section II discusses related work. Section III presents our research methodology. Section IV introduces *ROUTE*. Section V presents the experiments to validate the feasibility of the framework. Section VI discusses the framework. Section VII concludes the paper and outlines future work directions.

II. Related Work

The aim for smarter transportation management has been growing together with the increasing awareness of climate change risks and the impacts caused by everyday activities. However, it is critical to provide citizens with effective access to public transportation to convince and keep them using it [2]. Even more important, transportation alternatives shall be suited for all citizens, even those with accessibility hinders or specific needs [3]. Moreover, different (smart) cities have their own infrastructure (e.g., IoT sensors network) and regulations, thus the diverse (smart) cities stakeholders (e.g., municipality, city planners) may have different concerns and goals on their cities (e.g., promoting a transportation mean over the others). Lastly, problems of traditional public transportation such as delays and a certain rigidity in available alternatives are typically recognized as discouraging factors when compared to private transportation [10]. However, as we discuss in the following, traditional multimodal travel planners do not show a proper degree of support to dynamically deal with the beforementioned needs and limitations.

In our previous studies [8], [11], we used and evaluated a representative selection in the ecosystem of multimodal journey planners to verify their offered support for planning configuration and journey monitoring. Our final findings show that nearly half of them do not provide any configuration and monitoring support, whereas the remaining ones handle some mobility resources (i.e., closed parking spaces, bus timetable updates) and/or provide alerts and notifications about, e.g., traffic status (see [8], Fig. 8). Further, the main data source providing this information is users alerts and signaling. Currently, there is no standard way to include and/or modify city mobility policies and data into journey planners. On the contrary, these are hard-coded and not easy to (dynamically) change, thus making journey planners too strict. Moreover, in [8], the authors propose a model-driven solution to cater to mobility-relevant data (e.g., bus lines, bike-sharing stations, etc.) that are amalgamated to create suitable and customized multimodal plans. Consistently with this contribution, that paper also argues that finer-grained/local information about transportation alternatives is vital to support effective multimodal travel planning and a wider users engagement. However, the authors do not specify any architecture to manage such sources of information, nor how to possibly integrate mobility-relevant information with automated adaptation. From a wider perspective, Mandžuka's work [10] copes with the use of multimodal planners for cross-borders travels. In such scenarios, the main underlying issue is the potential lack of complete data (e.g. those for other countries than the origin one) to propose optimal door-to-door plans. The author proposes an architecture for distributed planning that leverages open interfaces to propagate a cross-borders request towards the corresponding country planners and then integrates the various alternatives received in response. As a matter of fact, traveling across smart cities can be reduced as crossing borders with respect to available fine-grained transportation information. Therefore, a multimodal journey across different cities necessarily requires distributed planning.

Usually, journey planning cannot be considered in isolation; instead, it should be immersed in a smart city ecosystem, i.e., including goods transportation, security services, etc. The systematic literature review discussed in [12] provides an overview of recent research efforts dealing with AI contributions to redefine transport, providing a user-centered technology that "understands" and "satisfies" the human user. As discussed, IoT provides enabling technologies for sensing, analysis, and provision of smarter services, e.g., customized on users' preferences. For example, travelers could have other goals than just moving from an origin to a destination point in a smart city; notably, tourists may wish to get an adequate journey arrangement touching selected points of interest and considering parameters as available time, kind of attractions, weather, and so forth [4], [5]. In this respect, in [4] the authors propose a general framework to benefit from IoT and artificial intelligence (AI/ML) solutions for providing tourists with smart travel plans, and in particular recommending points of interest. Similarly, Bin et al. [6] present a route recommendation method for tourists: based on the travelers' profiles and their match with the historical data from previous travelers the method derives the highest-ranked journeys. These studies testify the potentials of adequately managing local information and highlight the inadequacy of traditional travel planners.

Research efforts have also been targeting multimodal and adaptive journey planning [7]. In fact, a fundamental capability of multimodal travel planning is adaptation: especially due to delays, a plan based on several transportation means could completely fail and make, e.g., the travel time unacceptable. In this respect, research has been devoted to simulating traffic scenarios for predicting flow modifications and anticipating adaptation needs for ongoing journeys. In particular, the works in [2] and [9] leverage agent-based platforms to simulate traffic flows: the former targets the evaluation of intervention measures over traffic infrastructures and could be used by policymakers, planners, and service operators to understand potential consequences of manipulations on transportation assets; the latter deals with adaptive traffic management by proposing a simulation-based planner that optimizes load balance on streets. Other works use AI/ML solutions to improve the planning based on specific objectives: notably, Bustos et al. [13] propose a deep learning approach to locate city areas subject of risks for pedestrians and consequently introduce heuristics to alleviate the risks through traffic interventions.

To summarize, the related studies present approaches for enabling trip planning and adaptation services. However, no work aims at developing the new generation of travel planners by involving different actors in the dynamic smart cities ecosystems, taking the software engineering perspective.

III. Research Method

To design the architecture of the Smart Journey Planning Framework, we applied the general model proposed by Hofmeister *et al.* [14]. The authors devised the model by analyzing and comparing five industrial software architecture design methods. The activities of the model are summarized below and also shown in Figure 1 [14]:

Architectural Analysis: this activity aims at defining the problem that the framework architecture should solve. Specifically, the main output of this activity is a set of *Architecturally Significant Requirements* (ASR), which influence the framework architecture through analyzing the relevant architectural concerns and context [15]. According to the IEEE 1417, *architectural concerns* are interests that are important to a system's stakeholders or relevant to the system's development or operations [16]. The *system context* determines the settings and circumstances that influence the system. The context includes the business goals of the organization developing the system, the current state of the technology, and the system's development and operations processes [15], [16]. To this aim, we first described the main system contextual aspects, together with a set of relevant scenarios and architectural concerns. Based on these, we then identified a set of ASR. The results of this activity are reported in section IV-A.

Architectural Synthesis: this activity aims at identifying candidate architectural solutions that can meet the formulated ASR. The candidate solutions demonstrate design decisions concerning the framework's architecture and include information about the rationale behind those decisions. To perform this activity, we started by defining the conceptual model of our framework. Then, we further realized a feature model for our framework that has been used as a driver for modeling the proposed architecture and its corresponding process. All these artifacts are described in section IV-B.

Architectural Evaluation: this activity measures the candidate architectural solutions against the set of ASR. Specifically, to this aim, we performed both a scenario-based evaluation, reported in section IV-C and the evaluation of the prototype we realized, described in section V.

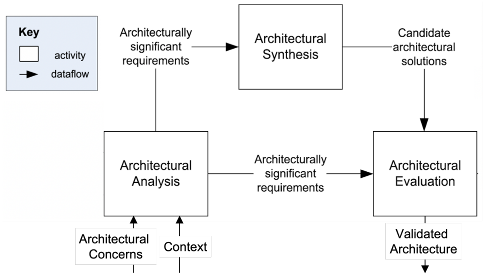


Fig. 1: The architecture design activities [14]

IV. The *ROUTE* Framework

In this section, we provide details about how we performed each activity of the model proposed by Hofmeister *et al.* [14].

A. Architectural Analysis

Context. In the following, we define the developmental and operational circumstances of multimodal journey planners according to the related work discussed in Section II, together with a set of relevant scenarios that we want our framework to allow. The main context-related aspects are:

C1: Most travel planners do not consider *mobility constraints* imposed by authorities when generating trip paths. Examples of such constraints include: (a) temporary closing a city center due to reconstruction works; (b) limiting the number of people who can gather concurrently to attend an outdoor event due to the current pandemic. Although multiple global journey planners are available as open-source (e.g., Rome2Rio[2], Open Trip Planner[3]), software developers are needed to customize them to define mobility constraints [8].

C2: Most travel planners do not provide *customized service based on user categories or preferences*. For instance, (a) while traveling to a place of interest, a tourist might be interested in passing through places where events are ongoing in a city, or (b) through non-crowded streets.

C3: Most travel planners do not consider *real-time data about environments*. Consequently, they do not provide accurate travel planning adaptation services. For instance, (a) the planners do not consider the number of people walking in a street. Thus, people can find themselves in crowded places violating regulations related to the pandemic. (b) Planners do not consider the noise or pollutants levels in streets, which travelers might want to avoid.

In Table I, we report a set of possible scenarios that we defined to address the shortcomings stemming from the literature, and those scenarios we aim to enable by our framework. Each scenario is described according to the template adapted from [17]. Notably, we specify (i) the *Actor* involved in the scenario, as some entity (e.g., a human, a computer system) that generates a stimulus; (ii) the *Stimulus*, i.e., an event arriving from the actor or the environment; (iii) the *Artifact*, i.e., the system or part of it that is required to be available to deal with the given stimulus; (iv) the *Response*, as the activity undertaken as the result of the management of the stimulus.

Architectural Concerns. Several concerns are relevant when engineering *ROUTE*, including performance, scalability, security, privacy, usability, the accuracy of the provided travel plans, availability, and reliability—considering the above concerns when engineering *ROUTE* requires evolving the framework through multiple iterations. In this first iteration, we mainly focus on performance and scalability, while we plan to address the remaining concerns in future iterations. After analyzing the relevant architectural concerns and context

[2] https://www.rome2rio.com [3] https://docs.opentripplanner.org/en/latest/

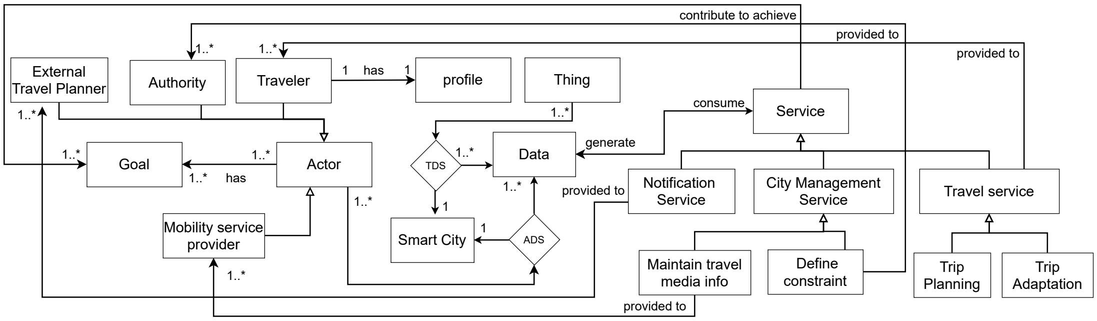


Fig. 2: Conceptual model of the Framework (described in UML)

ID	*Scenario*
SC1	(i) *Actor*: the Municipality of city A. *Stimulus*: the municipality decides to close the city center during the next weekend due to reconstruction work. *Artifact*: the mobility administration application. *Response*: the city center is considered as an inaccessible place to travelers. (ii) *Actor*: a traveler. *Stimulus*: the traveler wants to get guidance to reach a shop close to the city center, during the weekend. *Artifact*: the travel planning application. *Response*: the traveler is recommended to go through a route that does not comprise the city center.
SC2	*Actor*: Lorraine is a tourist visiting city A. *Stimulus*: Lorraine wants a recommended travel path towards the city A museum, passing through places where cultural events are currently organized. *Artifact*: the travel planning application. *Response*: Lorraine is recommended a path that includes three places where cultural events are organized into her way towards the museum.
SC3	*Actor*: Mark feels stressed after a long working day. *Stimulus*: he wants to be recommended a quiet walking path towards his home. *Artifact*: the travel planning application. *Response:* Mark is recommended an uncrowded walking path.
SC4	*Actor:* the travel planning application provides support to a traveler to reach a store. *Stimulus*: an accident that results in closing a street within the recommended travel path. *Artifact:* the smart travel planning infrastructure. *Response*: the smart travel planning infrastructure updates the travel planning application about the accident and provides it with an alternative path that the traveler can use to reach her/his destination.
SC5	*Actor*: the bus company of city A. *Stimulus*: The company wants to notify the travel planners about its updated bus paths and timetables. *Artifact*: the mobility administration system. *Response*: users of travel planners will be provided with up-to-date plans.

TABLE I: Scenarios

ID	*Requirement*
FR1	The framework should enable authorized parties to define constraints on travel routes or places of interest.
FR2	The framework should maintain and analyze up-to-date information about the city environment and activities and the constraints defined by the city authorities.
FR3	The framework should enable travelers to express their preferences.
FR4	The framework should enable authorized parties to update the information about travel media (e.g., schedules, paths, locations, prices).
FR5	The framework should provide travelers with travel plans considering the travelers' preferences, current status of the city, and the constraints defined by the city authorities.
FR6	The framework should adapt the travel plans provided to travelers to cope with the dynamic changes in the smart city, considering the travelers' current locations, preferences, and the constraints defined by the city authorities.
NFR1	The framework should be responsive when providing travel plans to an increasing number of travelers.

TABLE II: Architecturally Significant Requirements (ASR)

described above and the scenarios discussed in Table I, we identified the ASR reported in Table II, including functional requirements (FR) and non-functional requirements (NFR).

B. Architectural Synthesis

Conceptual Model. Figure 2 shows the conceptual model of our framework. An *actor* represents a category of users of the framework. We consider four categories of users as described in the following. An *authority* (e.g., a municipality or a police department) is an authorized organization whose *goals* include organizing traffic and enforcing a smart city's law. The framework supports authorities to achieve their *goals* by enabling them to define constraints and consider those constraints when providing services to other actors (see SC1 in Table I). A *constraint* is a restriction on movement. Authorities can define constraints of different types, including: (i) *reconstruction constraints* that restrict access to places or streets due to reconstruction works; (ii) *Covid-19 constraints* that limit the number of people who can gather concurrently in a place; (iii) *cultural event constraints* that restrict access to specific routes or places due to cultural events (e.g., carnivals).

A *traveler* is a user whose *goal* is to move around a city, being supported and informed about how to reach destinations. The traveler has a *profile* recording her/his previous trips and preferences. For example, a traveler could have preferences about the types of transportation media, the traffic conditions (e.g., uncrowded), and the types of activities/events organized in the places located between trips' origins and destinations (see SC2 and SC3 in Table I). The framework supports travelers in achieving their goals by providing two types of services: *trip plan generation* and *adaptation*. In the former,

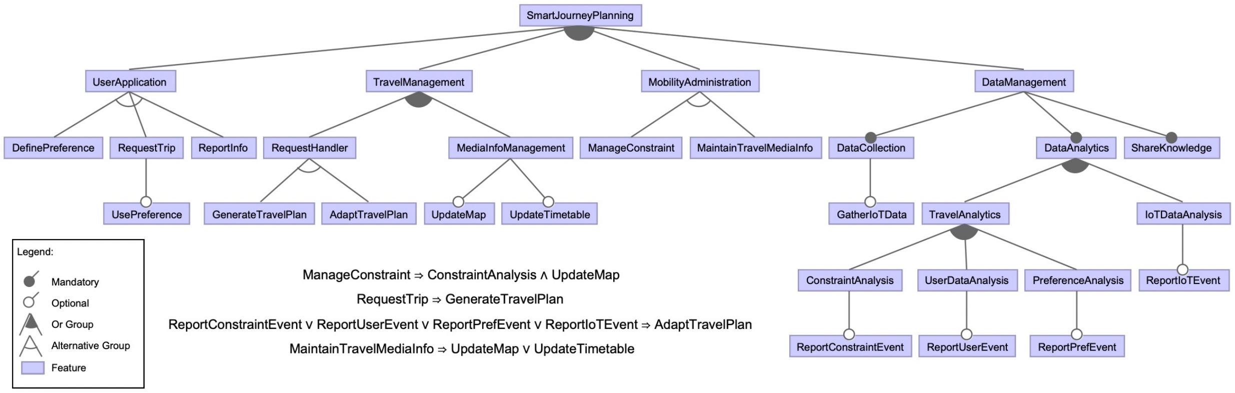


Fig. 3: Feature Model Diagram

the framework recommends a trip plan starting from the trip's origin to the destination. In the latter, the framework automatically redirects the traveler to the destination through an alternative route due to changes in the previously proposed route's status (see SC4 in Table I). When providing both types of services, the framework considers the travelers' preferences, the current status of routes, and the constraints defined by the authorities of the smart cities. The current status of routes can be determined based on data reported by travelers or collected by *things* (i.e., IoT devices and objects such as cameras and sensors). These are represented by the ternary relation Actor reports Data about a Smart City (ADS) and Thing report Data about a Smart City (TDS), respectively. For instance, a user reports a traffic accident that blocked a route in the smart city. Another example is when the analysis of the data collected sensors installed in a street show that the street is crowded and has a high-level of noise.

A *mobility service provider* (e.g., a bus company) is an organization whose *goals* include providing accurate mobility services to travelers. The framework supports mobility service providers to achieve that goal by enabling them to keep the paths and timetables of their travel media updated and consider this information when providing services to travelers (see SC5 in Table I). Finally, an *external travel planner* is a third-party application (e.g., Google Maps) whose goals include providing trip planning services to travelers. We plan to evolve the framework to support such applications to achieve that goal by exposing services that the applications can consume to enrich their travel planning services.

Feature Model. Figure 3 shows the Feature Model (FM) we defined for our framework. The nodes in the FM can be associated with and-or relations in addition to cross-tree constraints. The FM presents the features that our framework should provide, according to the context, scenarios, and requirements previously highlighted. We developed the FM using the *FeatureIDE* framework [18], [19], an Eclipse-based IDE that supports all phases of feature-oriented software development. A *SmartJourneyPlanning* framework integrates a *UserApplication*, a *TravelManagement* module, a *MobilityAdministration* module, and a *DataManagement* module. The *UserApplication* allows users to *DefinePreference*s, *RequestTrip*s, optionally taking into account the *UsePreference*s (**FR3**), or *ReportInfo* in a crowd-sensing manner. The *TravelManagement* module is that providing the journey planning and adaptation functionalities (**FR5** and **FR6**). To this aim, the TravelManagement module should handle incoming requests (*RequestHandler* feature) and *GenerateTravelPlan* or *AdaptTravelPlan*. Moreover, it should keep up-to-date information (*MediaInfoManagement* feature) by means of *UpdateMap* and *UpdateTimetable* functionalities (**FR2** and **FR4**). The *MobilityAdministration* module refers to the features provided to authorities and mobility service providers allowing them to *ManageConstraint*s and *MaintainTravelMediaInfo* (**FR1** and **FR4**), respectively. Lastly, different data from different sources must be collected, processed, and shared in our framework. To this aim, the *DataManagement* feature groups the *DataCollection*, *DataAnalitics*, and *ShareKnowledge* features (**FR2** and **FR4**). In addition to the actors in the system, data can come from an IoT network installed in the city (*GatherIoTData* feature). Since changes in the environment might impact the ongoing trips, the framework should be able to perform *TravelAnalytics* and *IoTDataAnalysis* (**FR2** and **FR4**). Specifically, the travel analytics comprises *ConstraintAnalysis*, *UserDataAnalysis*, and *PreferenceAnalysis*. Indeed, every change in the defined constraints and users preferences, as well as changes in the environment reported by users or IoT devices might trigger events leading to the adaptation of the ongoing trip plans (**FR6**). Triggering events is represented by the *ReportConstraintEvent*, *ReportUserEvent*, *ReportPrefEvent*, and *ReportIoTEvent* features.

Additionally, we defined some cross-tree FM constraints[4] enriching the semantic of the FM. A first FM constraint forces the *ConstraintAnalysis* and *UpdateMap* features to be performed any time a new constraint is defined by the authority. A second FM constraint forces the *GenerateTravelPlan* feature any time a *RequestTrip* arrives. A third FM constraint specifies that the *AdaptTravelPlan* is triggered by any of the report event features. A fourth FM

[4] To distinguish between the constraints from the authorities and those on the Feature Model (FM), from here on we call the latter *FM constraints*.

constraint says that the *MaintainTravelMediaInfo* implies an *UpdateMap* or an *UpdateTimetable*. We used the above FM as a basis to drive the design of the *ROUTE* architecture.

Architecture. Based on the analysis of the formulated scenarios and the FM, we identified some candidate architectures that meet the ASR to different extents. Due to the limited space, we briefly discuss three candidate architectures. The first candidate is an architecture that adopts the client-server model, where the data management and travel management features in Figure 3 are provided centrally by one or more servers that can auto-scale based on the needs. The second candidate is a (micro) service-based architecture where the leaf features in the FM are wrapped into (micro) services, and orchestrators are designed at the upper levels of the FM. The third candidate is a multi-tier architecture where the features under the first level of the FM are grouped into independent execution structures (i.e., tiers). Concerning the second candidate, in general, (micro-) service-based architectures scale well and provide better utilization of resources across the Edge-Cloud continuum. However, decomposing the functionalities of existing travel planners into (micro-)services and enacting and managing those services at a large scale in distributed settings are complex tasks [20] we will investigate in our future work (see Section VI). Therefore, we chose to implement the third candidate architecture for our framework because the tiers can be deployed in distributed resources, resulting in better performance, scalability, and efficient utilization of resources compared to the first candidate architecture.

Figure 4 shows the multi-tier architecture of our framework. It comprises five tiers that exchange data through an Enterprise Service Bus (ESB). An ESB is a secure, performing, scalable, and reliable communication media that supports interoperability between the architecture's tiers [21], [22].

1) Traveler tier. This tier comprises four components responsible for enabling travelers to interact with the framework (**FR3**). The *user profile* records users' preferences and previous trips. Users can have preferences concerning: (i) the characteristics of the routes that lead to the destinations they aim for (e.g., see SC2 and SC3 in Table I); (ii) the trips' modality (e.g., walking, by bus, or by train). The *trip requester* enables travelers to submit trip planning requests. The *trip guidance manager* parses the trip plans produced by the trip planning and adaptation tier and accordingly provides instructions that facilitate reaching the trips' destinations. The *data reporter* enables travelers to report data about places or routes. For instance, a traveler reports that a street is crowded, another traveler reports that an accident has blocked a street.
2) Mobility administration tier. This tier comprises two components responsible for enabling authorities to interact with the framework (**FR1** and **FR4**). The *dashboards* provide an overview of the current status of the smart city, including the active constraints, the violated constraints, and the events that impact the movement in the city (e.g., traffic accidents). That knowledge is retrieved from the data tier. The *editors* enable authorities to manage constraints on movement. This includes defining, updating, or canceling constraints.
3) IoT tier. This tier comprises four components responsible for connecting IoT things and collecting their data (**FR2**). The *things registry* comprises connected things, the locations where the things are installed, and the things' connectivity status. The *things manager* monitors the things and updates their connectivity status in the things registry. The *data aggregator* periodically aggregates and formats the things' data. Finally, the *data publisher* periodically publishes the aggregated data to the data tier through the ESB.
4) Data tier. This tier comprises three components responsible for deriving and sharing knowledge to facilitate and manage movement in a smart city (**FR2**). The *knowledge base* component represents the container of the context concerning movement in the city. The knowledge base has the following sub-components:
   a) The trip requests repository stores the requests made by travelers, the related preferences, and the status of those requests (i.e., handled or pending).
   b) The big data repository stores the data shared by the IoT tier and the data reported by travelers (e.g., traffic accidents).
   c) The constraints' repository stores the restrictions on movements defined by authorities and their status (i.e., active or inactive).
   d) The city live data repository stores up-to-date knowledge about the status of city routes, such as the number of people walking in streets, the air pollution and noise levels in the city neighborhoods. Such knowledge is derived by analyzing the data reported by travelers or IoT things.
   e) The trips repository stores information about requested, ongoing, and completed trips. We store the map identifiers of the involved routes and places for each trip (see the maps repository in the travel planning and adaptation tier).
   f) The activities repository stores information about the activities organized in the city. Specifically, it stores activities' types (e.g., cultural events), when and where the activities are organized (see SC2 in Table I).
   g) The event repository stores events representing violations of movement constraints or user preferences. For instance, an event is triggered automatically when the maximum number of people allowed to gather in a square is reached. Another event is automatically triggered when the travel path recommended to the traveler in SC3 in Table I is no longer quite, e.g., due to road traffic noise.

   The *data analytics* component analyzes the data stored in the knowledge base to derive knowledge about the

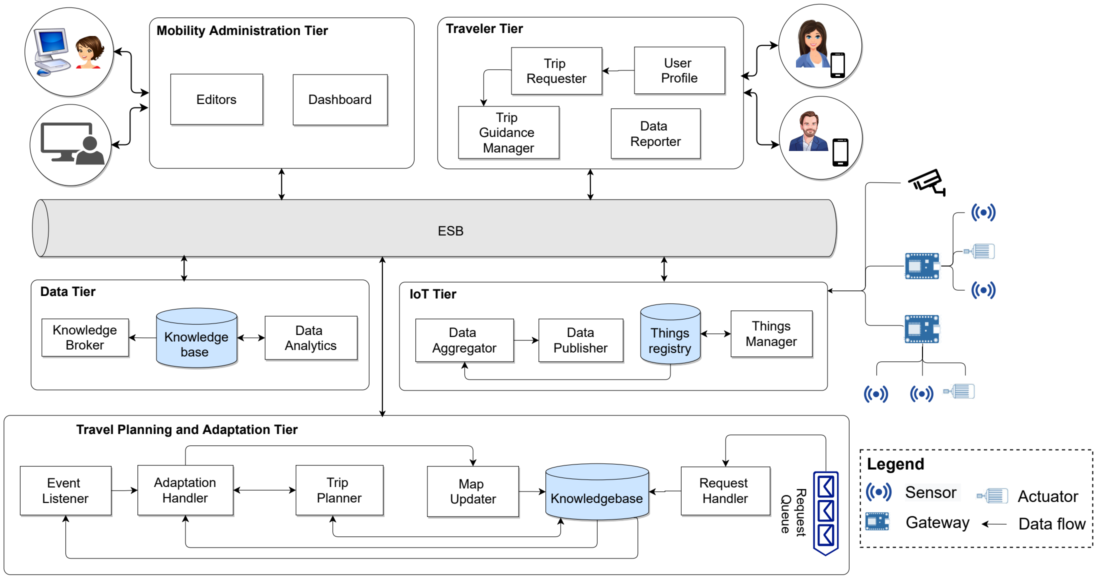


Fig. 4: The multi-tier architecture of the *ROUTE* framework.

movement in the city. The data analytics component has the following sub-components:

a) The city data analyzer analyzes the data in the big data repository and updates the city live data repository with knowledge about the status of city routes (e.g., noise level).
b) The constraints' analyzer analyzes active constraints and evaluates if they are violated, considering the knowledge stored in the city live data. For instance, in SC1 in Table I, in the weekend, the constraint analyzer requests the map updater component (in the planning and adaptation tier) to remove the vertex representing the city center and all edges directly leading to it from the city map graph, so the trip planner does not consider them when generating travel plans. When a constraint is violated, the constraint analyzer triggers an event automatically. The event is stored in the event repository and pushed to the planning and adaptation tier to adapt ongoing impacted trips.
c) The preference analyzer evaluates if the current travelers' preferences are violated and triggers events accordingly. For instance, in SC3 in Table I, the preference analyzer triggers an event if the path recommended to the user is no longer quiet.

Finally, the *knowledge broker* shares the knowledge stored into the knowledge repository with the other tiers through the ESB.

5) Travel planning and adaptation tier. This tier comprises six components responsible for generating and adapting travel plans (**FR5** and **FR6**). The requests to generate or adapt travel plans are queued in the *request queue*. A request also includes the requester's preferences when they are specified. The *knowledge base* component has the following sub-components:

a) The city map graph repository comprises supported cities' maps. A map's graph is composed of vertices representing coordinates of places and edges representing the routes among the places.
b) The travel media info repository hosts information concerning travel media, e.g., timetables and paths.
c) A cache data structure that stores temporarily active constraints, trip requests and related preferences, ongoing trips and the corresponding travel plans, and the events reported by the data tier and the impacted ongoing trips. This cache is needed for performance purposes, so the travel planning and adaptation tier avoids retrieving data from the data tier every time the former needs to generate or adapt travel plans.

The *map updater* updates the maps in the city graph map repository based on the context. Specifically, for events reflecting opening/closing routes or places in a city, the map updater updates the city map by adding/removing the vertexes and edges corresponding to those entities. The *trip planner* generates travel plans considering the travelers' preferences, the city map graph, the constraints defined by the authorities of the city, and the up-to-date status of the movement paths in the smart city. The *adaptation handler* adapts the ongoing trips impacted by the events reported by the data tier. For this purpose, the travelers' current locations are considered when regenerating their travel plans.

Process. Figure 5 shows the process of generating and adapting travel plans in a smart city. The IoT tier continuously collects data from IoT devices and objects installed in the city and shares the data with the data tier. When a mobility service provider updates its travel media's timetables or paths, the knowledge base at the travel planning and adaptation tier is updated. Likewise, when an authority defines a constraint, the knowledge bases of both the data and travel planning and adaptation tiers are updated.

The trip planning process has three phases: pre-planning, planning, and post-planning. In the *pre-planning phase*, the data tier analyzes the constraints in its knowledge base and instructs the travel planning and adaptation tier to update the map of the city accordingly. For instance, in SC1 in Table I, the city center is removed from the map city during the weekend.

In the *planning phase*, a user requests a travel plan by specifying the trip's origin and destination and optionally her/his preferences. The request is stored in the knowledge bases of the travel planning and adaptation tier and the data tier. The former tier then generates a set of candidate travel plans that lead the traveler from the origin to the destination. In the *post-planning phase*, the travel planning and adaptation tier filters the candidate plans based on the traveler's preferences, the current status of the paths, and the defined constraints. For instance, in SC3 in Table I, the planner ranks candidate travel plans based on the number of people walking in the corresponding streets. Accordingly, the planner recommends to the traveler the least crowded path.

Despite filtering unsuitable candidate plans during the pre-planning and post-planning phases, ongoing plans might violate the defined constraints or the travelers' preferences due to the dynamicity of the smart cities. For instance, the number of people gathering in a square in a city might increase to the limit specified by the city's authorities after the planner recommended travel plans comprising the square to travelers. Additionally, travelers can report data about the places' status (e.g., an accident blocking a street). To cope with such dynamicity, the data tier continuously analyzes the data reported by travelers and IoT devices, and it evaluates if the current travelers' preferences or the active constraints are violated. When a violation is detected, the data tier triggers an event and shares with the travel planning and adaptation tier the impacted ongoing trips. The latter tier adapts the corresponding travel plans by regenerating them, considering the current impacted travelers' locations, as described above.

C. Architectural Evaluation

According to Hofmeister *et al.* [14], architectural solutions are evaluated with respect to the ASR that they address. The procedure of architectural evaluation may include model-based analysis, simulation, prototyping, and discussion of user scenarios. This section evaluates the proposed multi-tier architectural solution employing a scenario-based evaluation. Specifically, we verified that the scenarios in Table I could be successfully modeled as feature configurations of the FM in Figure 3, meaning that they belong to the set of *valid* feature configurations allowed by the FM and its constraints. FeatureIDE allows the definition of configurations through both the automatic and manual selection (and deselection) of features from the FM. This means that the FeatureIDE configurator can support the modeler by automatically selecting or deselecting features that must or must not, respectively, be part of the configuration the modeler is defining. Lastly, FeatureIDE automatically builds the configurations and determines if they are valid or not.

Figure 6 shows the XML representation of the FM configuration corresponding to the scenario **SC1** in Table I. Specifically, the *manual* selection of the features *ManageConstraint* (line **17**) and *RequestTrip* (line **6**), corresponding to the Stimulus of SC1(i) and SC1(ii), respectively, triggered the *automatic* selection and deselection of the remaining features, by the configurator. Given that each feature corresponds to a boolean variable and the semantics of the FM is captured as a *propositional formula*, each valid feature configuration is a satisfying valuation of this formula. Indeed, the textual configuration in Fig. 6 corresponds to a conjunction of FM constraints of the propositional formula. For instance, line **11** corresponds to the FM constraint in equation (1):

$$RequestTrip \implies GenerateTravelPlan \tag{1}$$

Indeed, *GenerateTravelPlan* has been automatically selected because of the constraint in equation (1), which has been triggered after that *RequestTrip* has been manually selected. Another example refers to line **14** in the textual configuration in Fig. 6. *UpdateMap* has been automatically selected after the manual selection of *ManageContraint* because of the FM constraint in equation (2):

$$ManageConstraint \implies ConstraintAnalysis \wedge UpdateMap \tag{2}$$

Eventually, the feature configuration corresponding to SC1 has been assessed to be valid by the FeatureIDE configurator. With respect to the ASR, the valid configuration of SC1 addresses the requirements **FR1**, **FR4**, and **FR5**. We performed the same evaluation for the remaining scenarios in Table I. However, for lack of space, we report their corresponding graphical configurations in our online material.[5]

V. Validation

To validate the feasibility of our framework and evaluate its performance and scalability (**NFR1**), we developed a first prototype and used it to run experiments. To generate and adapt travel plans, we integrated the Open Trip Planner (OTP),[6] in the travel planning and adaptation tier. We used the OTP because it is a multimodal and open-source planner that provides plans combining bicycle, transit, car, and pedestrian segments by exploiting the OpenStreetMap[7] and GTFS[8] data. In the *first experiment*, we evaluated the performance and scalability of the travel planning and adaptation tier when no constraints on movement were specified. In the *second experiment*, we simulated the performance and scalability of the framework when dynamically defining constraints specifying that three routes in a smart city are inaccessible. The data tier was developed to analyze the defined constraints automatically and to instruct the travel planning and adaptation tier to update the city map accordingly, as described in the pre-planning sub-process presented in Section IV. We conducted both experiments in both centralized and distributed deployment

[5] https://bit.ly/31zR79L.
[6] https://www.opentripplanner.org/
[7] https://bit.ly/3odWTFQ
[8] https://gtfs.org/getting-started/

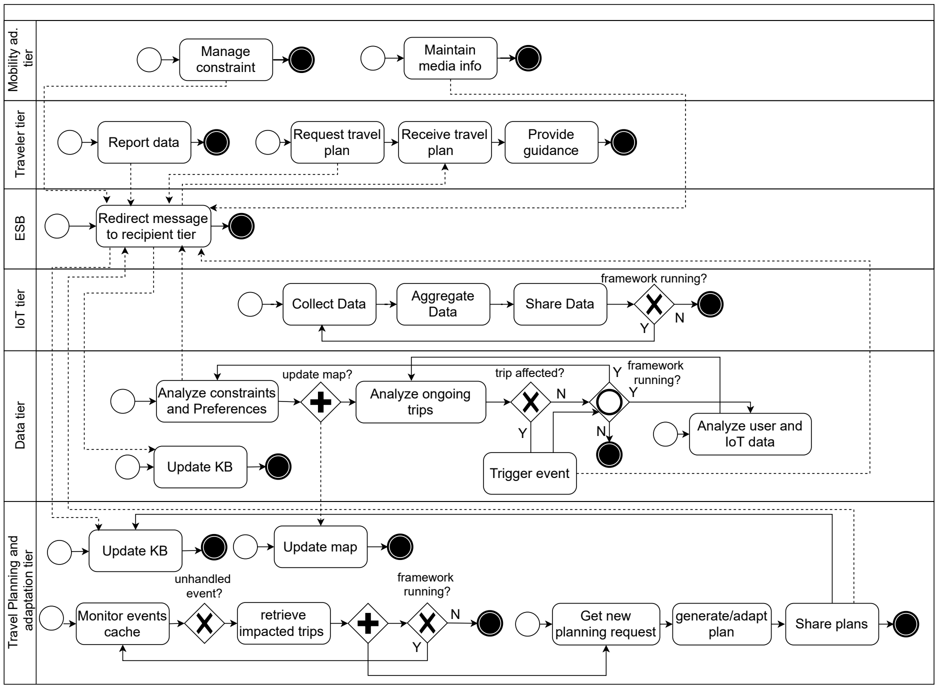
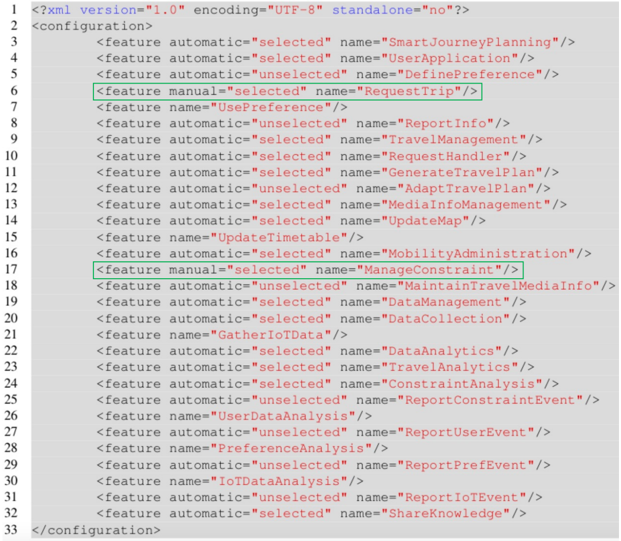


Fig. 5: The process of generating and adapting travel plans



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configuration>
        <feature automatic="selected" name="SmartJourneyPlanning"/>
        <feature automatic="selected" name="UserApplication"/>
        <feature automatic="unselected" name="DefinePreference"/>
        <feature manual="selected" name="RequestTrip"/>
        <feature name="UsePreference"/>
        <feature automatic="unselected" name="ReportInfo"/>
        <feature automatic="selected" name="TravelManagement"/>
        <feature automatic="selected" name="RequestHandler"/>
        <feature automatic="selected" name="GenerateTravelPlan"/>
        <feature automatic="unselected" name="AdaptTravelPlan"/>
        <feature automatic="selected" name="MediaInfoManagement"/>
        <feature automatic="selected" name="UpdateMap"/>
        <feature name="UpdateTimetable"/>
        <feature automatic="selected" name="MobilityAdministration"/>
        <feature manual="selected" name="ManageConstraint"/>
        <feature automatic="unselected" name="MaintainTravelMediaInfo"/>
        <feature automatic="selected" name="DataManagement"/>
        <feature automatic="selected" name="DataCollection"/>
        <feature name="GatherIoTData"/>
        <feature automatic="selected" name="DataAnalytics"/>
        <feature automatic="selected" name="TravelAnalytics"/>
        <feature automatic="selected" name="ConstraintAnalysis"/>
        <feature automatic="unselected" name="ReportConstraintEvent"/>
        <feature name="UserDataAnalysis"/>
        <feature automatic="unselected" name="ReportUserEvent"/>
        <feature name="PreferenceAnalysis"/>
        <feature automatic="unselected" name="ReportPrefEvent"/>
        <feature name="IoTDataAnalysis"/>
        <feature automatic="unselected" name="ReportIoTEvent"/>
        <feature automatic="selected" name="ShareKnowledge"/>
</configuration>
```

Fig. 6: FM configuration for scenario 1 (SC1) in Table I

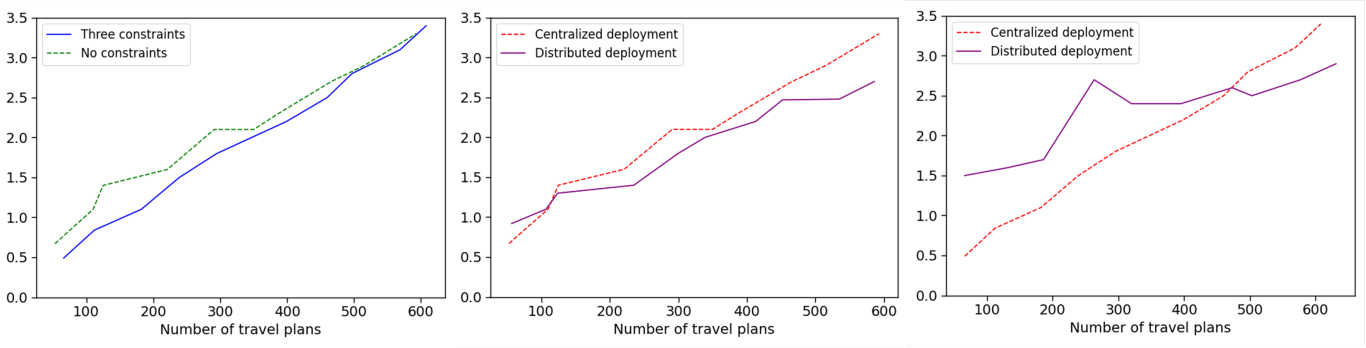
settings. The approach was deployed on a laptop using 4 core CPU running at 2.2GHz with 16 GB memory in the centralized deployment settings. In the distributed deployment settings, the approach was deployed on the laptop and a Cloud node using 8 CPU Cores, 640 GB Storage, with 32 GB memory.

Figure 7 (a) shows the time for handling trip requests when no constraints are defined and when three routes are inaccessible, in centralized settings. We simulated requests with origins and destinations in the vicinity of the inaccessible routes. Figure 7 (b) shows the total time for handling trip requests in both deployment settings where no constraints are defined. Finally, Figure 7 (c) shows the total time for handling trip requests in both deployment settings where three constraints are defined. The results show that when the number of requested plans increase, the framework seems to perform and scale well in both centralized and distributed deployment settings. As can be noted in Figure 7 (a), the time for handling requests, when the constraints are defined, is less than the time needed when no constraints are defined because the planner has fewer routes to consider when planning in the former case. Moreover, as can be noted in the Figures 7 (b) and (c), the framework performs and scales better when deployed in distributed settings when the numbers of concurrent generated plans exceed 150 and 480 requests, respectively. Furthermore, it can be noted that for part of the results, the time for handling trip requests when the three constraints are defined is higher in distributed settings (Figure 7 (c)) than in the centralized settings (Figure 7 (a)). That is probably because the process that analyzes the constraints and the process that updates the map accordingly run on different machines. As future work, we plan to perform a more intensive evaluation of the framework where we benchmark both scalability and performance and include additional quality attributes such as security, privacy, and reliability.

VI. Discussion

Travel planners should evolve to meet the rapid development of our smart cities. In our framework, we integrated an open-source travel planner. However, many people use commercial or closed source travel planners (e.g., Google Maps[9] and Sygic[10]). Such planners would enrich their services when they integrate with smart cities infrastructures. For instance,

[9] http://maps.google.com/ [10] https://www.sygic.com/



(a) Centralized deployment

(b) No constraints in both deployment settings

(c) Three constraints in both deployment settings

Fig. 7: The time for handling trip planning requests

a service can notify the travel planners about traffic accidents detected automatically by smart cameras and the traffic jams caused. The planners no longer require users to share their personal data (e.g., the current driving speed) to calculate traffic jams by exploiting such services. Thus, users have better privacy. Additionally, the integration would enable the travel planners to consider the constraints defined by local authorities and the most updated travel paths and timetables of the different travel media in the cities.

Evolution scenarios can emerge in different application domains. For instance, self-driving vehicles will be widely used shortly. Those vehicles can perform vehicle-vehicle communications and are equipped with processing and storage capabilities to run smart software. Due to privacy concerns, users may decide to perform the travel planning and adaptation locally on their smart vehicles. Our framework supports users to achieve this goal by enabling the deployment of the travel planning and adaptation tier (e.g., as an application) on the users' smart vehicles. Another evolution scenario is the following. In the Police operation room, the operators need to see the location of each patrol, the personnel in the patrol, the mission the patrol performs, and the best route it can take to accomplish the mission. For this purpose, the Police needs to extend the functionalities of the travel planning system but require deploying those extensions on their own servers for security purposes. Based on the current design, it would be required to extend and deploy one or more tiers of *ROUTE* in the Police department servers. To better meet such special needs, some of the functionalities of *ROUTE* can be exposed as (micro-) services and provided in a service registry to be consumed by authorized parties. Exposing services would better support evolution scenarios through enabling service composition and extension to deliver new or customizable functionalities. The functionalities of the different tiers can be exposed as services. However, although decomposing the trip planning applications into (micro-) services can support the scalability and extendability of the application, it might also cause an overhead if those services are deployed on the same machine (e.g., server or smart vehicle). We plan to address these aspects in our future work.

The data provided by IoT things and users about their environments can be exploited by planners to generate accurate travel plans. However, engaging people in this process, validating the information they provide, and analyzing the huge and heterogeneous data collected by the IoT things can be challenging. For instance, people can describe events and their consequences differently. Therefore, advanced data analytics techniques are needed to analyze the data. Moreover, techniques are needed to engage people to report data and also evaluate the services provided by the framework, including the accuracy of the suggested plans. Furthermore, engineering intelligent IoT systems requires addressing novel challenges concerning e.g., the evolution and bias in data, continuous training, trustworthiness in the systems' decisions, and distributed systems' configurations. For this purpose, novel methods are required to engineer system and software architectures for intelligent systems [23]. We plan to address these issues in our future work.

VII. Conclusions and Future Work

ROUTE is a framework for customizable smart mobility planners. *ROUTE* better fulfills travelers' needs, local authorities, and mobility service providers than travel planning applications available in the marketplace. The framework comprises a multi-tier architecture, a process, and a first prototype that we developed and used to validate the framework's feasibility.

As future work, we plan to put the framework into practice by integrating real IoT things in two cities and evolving it by exposing (micro-) services and implementing advanced data analytics and machine learning techniques to analyze the data collected from travelers and IoT devices and sensors. Moreover, we plan to consider additional non-functional requirements such as security, privacy, reliability, and availability. Finally, we plan to extend the developed prototype and conduct a more extensive evaluation.

Acknowledgments

This work is partially financed by the Knowledge Foundation through the Internet of Things and People research profile (Malmö University, Sweden) and by the PON Cultural Heritage (AIM1880573), Italian National Project at GSSI.

References

[1] F. P. Appio, M. Lima, and S. Paroutis, “Understanding smart cities: Innovation ecosystems, technological advancements, and societal challenges,” *Technological Forecasting and Social Change*, vol. 142, pp. 1–14, 2019.

[2] L. Yfantis, S. Stebbins, I. Gerostathopoulos, T. Djukic, J. Casas, D. Garcia, M. Kamargianni, and M. Chaniotakis, “A software-agnostic agent-based platform for modelling emerging mobility systems,” in *2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 2021, pp. 1–6.

[3] M. Sourbati, “Age and the city: The case of smart mobility,” in *Human Aspects of IT for the Aged Population. Technology and Society*, Q. Gao and J. Zhou, Eds. Cham: Springer International Publishing, 2020, pp. 312–326.

[4] W. Wang, N. Kumar, J. Chen, Z. Gong, X. Kong, W. Wei, and H. Gao, “Realizing the potential of the internet of things for smart tourism with 5g and ai,” *IEEE Network*, vol. 34, no. 6, pp. 295–301, 2020.

[5] X. Guo, Y. Wang, J. Mao, Y. Deng, F. Chan, and J. Ruan, “Towards an iot enabled tourism and visualization review on the relevant literature in recent 10 years,” *Mobile Networks and Applications*, 08 2021.

[6] C. Bin, T. Gu, Y. Sun, L. Chang, and L. Sun, “A travel route recommendation system based on smart phones and iot environment,” *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[7] M. Graczyk-Raczyńska, M. Olszewski, K. Gówczewski, and D. Derebecki, “Multimodal journey planners – annual international markets overview,” Available at https://parkinggetssmart.eu/lib/f60d36/PgS-annual-market-overview-2020---Multimodal-journey-planners-kpf7z3fr.pdf, Tech. Rep., 2020.

[8] A. Bucchiarone and A. Cicchetti, “A model-driven solution to support smart mobility planning,” in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, Copenhagen, Denmark, October 14-19, 2018*, 2018, pp. 123–132. [Online]. Available: https://doi.org/10.1145/3239372.3239374

[9] I. Gerostathopoulos and E. Pournaras, “Trapped in traffic? a self-adaptive framework for decentralized traffic optimization,” in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2019, pp. 32–38.

[10] S. Mandžuka, “Providing multimodal traveler information cross-border journey planners approach,” in *New Technologies, Development and Application IV*, I. Karabegović, Ed. Cham: Springer International Publishing, 2021, pp. 665–672.

[11] M. De Sanctis, A. Bucchiarone, and A. Marconi, “Dynamic adaptation of service-based applications: a design for adaptation approach,” *J. Internet Serv. Appl.*, vol. 11, no. 1, p. 2, 2020. [Online]. Available: https://doi.org/10.1186/s13174-020-00123-6

[12] A. Nikitas, K. Michalakopoulou, E. T. Njoya, and D. Karampatzakis, “Artificial intelligence, transport and the smart city: Definitions and dimensions of a new mobility era,” *Sustainability*, vol. 12, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/7/2789

[13] C. Bustos, D. Rhoads, A. Solé-Ribalta, D. Masip, A. Arenas, A. Lapedriza, and J. Borge-Holthoefer, “Explainable, automated urban interventions to improve pedestrian and vehicle safety,” *Transportation Research Part C: Emerging Technologies*, vol. 125, p. 103018, 2021.

[14] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, “A general model of software architecture design derived from five industrial approaches,” *Journal of Systems and Software*, vol. 80, no. 1, pp. 106–126, 2007.

[15] H. Obbink, P. Kruchten, W. Kozaczynski, H. Postema, A. Ran, R. Dominick, Lutz Kazman, R. Hilliard, W. Tracz, and E. Kahane, “Report on software architecture review and assessment (sara),” Available at http://philippe.kruchten.com/architecture/SARAv1.pdf, Tech. Rep., 2002.

[16] R. Hilliard, “Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems,” *IEEE, http://standards. ieee. org*, vol. 12, no. 16-20, p. 2000, 2000.

[17] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[18] C. Kästner, T. Thüm, G. Saake, J. Feigenspan, T. Leich, F. Wielgorz, and S. Apel, “Featureide: A tool framework for feature-oriented software development,” in *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*, 2009, pp. 611–614.

[19] “Featureide.” [Online]. Available: https://featureide.github.io/

[20] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.

[21] R. Koh-Dzul, M. Vargas-Santiago, C. Diop, E. Exposito, and F. Moo-Mena, “A smart diagnostic model for an autonomic service bus based on a probabilistic reasoning approach,” in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*. IEEE, 2013, pp. 416–421.

[22] M. Luo, B. Goldshlager, and L.-J. Zhang, “Designing and implementing enterprise service bus (esb) and soa solutions,” in *2005 IEEE International Conference on Services Computing (SCC’05) Vol-1*, vol. 2. IEEE, 2005, pp. xiv–vol.

[23] J. Bosch, H. H. Olsson, and I. Crnkovic, “Engineering ai systems: A research agenda,” in *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*. IGI Global, 2021, pp. 1–19.