Cooperative Intelligent Transport Systems: Choreography-Based Urban Traffic Coordination

Marco Autili, Lei Chen, Cristofer Englund, Claudio Pompilio, and Massimo Tivoli

***Abstract*— With the emerging connected automated vehicles, 5G and Internet of Things (IoT), vehicles and road infrastructure become connected and cooperative, enabling Cooperative Intelligent Transport Systems (C-ITS). C-ITS are transport system of systems that involves many stakeholders from different sectors. While running their own systems and providing services independently, stakeholders cooperate with each other for improving the overall transport performance such as safety, efficiency and sustainability. Massive information on road and traffic is already available and provided through standard services with different protocols. By reusing and composing the available heterogeneous services, novel value-added applications can be developed. This paper introduces a choreography-based service composition platform, i.e. the CHOReVOLUTION Integrated Development and Runtime Environment (IDRE), and it reports on how the IDRE has been successfully exploited to accelerate the reuse-based development of a choreography-based Urban Traffic Coordination (UTC) application. The UTC application takes the shape of eco-driving services that through real-time eco-route evaluation assist the drivers for the most eco-friendly and comfortable driving experience. The eco-driving services are realized through choreography and they are exploited through a mobile app for online navigation. From specification to deployment to execution, the CHOReVOLUTION IDRE has been exploited to support the realization of the UTC application by automatizing the generation of the distributed logic to properly bind, coordinate and adapt the interactions of the involved parties. The benefits brought by CHOReVOLUTION IDRE have been assessed through the evaluation of a set of Key Performance Indicators (KPIs).**

***Index Terms*— Service choreographies, service composition, C-ITS, eco-driving, system of systems.**

I. Introduction

THE past centuries have witnessed a unique trend on urbanization with more and more people moving into large cities. By the year 2017, 4.1 billion people were living in urban areas and it is projected that by 2050 more than two-thirds of the world's population will live in urban areas [1]. Such a phenomenon brings benefits to the people with improved living standards, while also creating numerous challenges for the urban cities. One of the challenges is urban transport, as being demonstrated by the increasing mobility demand, congestion, pollution, and greenhouse gas (GHG) emissions due to e.g., urbanization, inefficient traffic control, and the fossil fuel vehicles still in use. In the EU, transport contributes with almost a quarter of EU's GHG emissions and is the main cause of air pollution in the EU cities [2]. What is even more concerning is that transport emissions are still on the rise, mostly due to the increasing demand on mobility.

On the other hand, technologies are advancing with rapid developments for connected automated vehicles, 5G, Internet of Things (IoT), etc., which lead to the emergence of Cooperative Intelligent Transport Systems (C-ITS) [3] and promises new innovative solutions to address above-mentioned transport challenges. C-ITS represent the evolution of connected and automated transport systems, where vehicles, road infrastructure, and traffic control systems are connected and cooperate for safe and efficient traffic. With many years research, pilot and standardization, C-ITS are at the stage of large scale implementation. This goes in parallel with the rapid development of connectivity, cloud computing and artificial intelligence (AI), and provides enormous opportunities to develop innovative applications for reducing the transport emissions.

C-ITS allow vehicles and transport infrastructure to interconnect, share information and use it to coordinate their actions. Coordination between vehicles and transport infrastructure creates a huge advantage in the form of enhancing awareness of the traffic participants. This enhancement will effectively contribute to the traffic safety, efficiency and driving comfort. Examples applications include those that help drivers to take decisions to avoid congestion, reduce traffic light stopping time, eliminate accidents from slippery or unsafe roads, etc. For reducing emissions, eco-driving is emerging where instead of choosing the shortest route or fastest route, the route with the minimal emissions is chosen. In addition, the route information is updated in real-time to reflect the infrastructure statuses such as traffic light, and the real-time traffic information. This, in combination with certain interaction methods to influence the drivers' behaviors, can potentially reduce the fuel consumption and emissions significantly, such as demonstrated by many earlier EU projects including COSMO, ecoMOVE, ecoDriver, and so on. Considering the trips being made every day in a city, such an application

Manuscript received June 20, 2020; revised December 10, 2020; accepted February 2, 2021. Date of publication February 24, 2021; date of current version March 31, 2021. This work was supported in part by the Italian Ministry of Economy and Finance under Project Innovating City Planning through Information and Communication Technologies (INCIPICT), Cipe Resolution under Grant 135/2012, in part by the SISMA National Research Project, which has been funded by the MIUR through the PRIN Program under Contract 201752ENYB, in part by the Connect PA National Research Project, funded by Regione Abruzzo under the POR FSER Abruzzo 2014-2020 program, and in part by the project System of Systems for efficient Emergency Response (SoSER) and urban mobility, which has been funded by the Swedish Innovation Agency Vinnova. The Associate Editor for this article was A. Bucchiarone. *(Corresponding author: Claudio Pompilio.)*

Marco Autili, Claudio Pompilio, and Massimo Tivoli are with the Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, 67100 L'Aquila, Italy (e-mail: marco.autili@univaq.it; claudio.pompilio@univaq.it; massimo.tivoli@univaq.it).

Lei Chen and Cristofer Englund are with the RISE Research Institutes of Sweden, 417 56 Götheborg, Sweden (e-mail: lei.chen@ri.se; cristofer.englund@ri.se).

Digital Object Identifier 10.1109/TITS.2021.3059394

is promising in the Urban Traffic Coordination (UTC) for contributing to the reduction of transport emissions.

While eco-driving is proven to be effective for emission reduction, its evolution with continuous addition of new data sources, new algorithms and rapid prototyping and validation is not trivial. C-ITS at their early stage provide connectivity, while seamless cooperation for rapid innovation remains a challenge. For example, traffic information and driving guidance services are widely available today as separated initiatives from different vendors and organizations. Similarly, embracing open data, many stakeholders provide access to their data through web-services such as road information and weather services. In addition, vehicles become a new type of sensor that can provide real-time information on e.g., road and traffic information. However, those closed initiatives only provide services from their own perspective and seldom coordinate with each other for even better optimized solutions taking transport systems as a whole. A traditional way for application development is to plan, collect and develop the application in a centralized way where a centralized controller coordinates such efforts. This usually requires proper technical and legal coordination between multiple involved parties including: transport authority, public transports, digital map providers, routing engines, weather forecast agencies, participated vehicles, research groups, and other service providers. Each involved party in this network provides and/or consumes data from other parties using different technologies, standards and data exploitation agreements with specific terms and conditions. This introduces multi-folds of complexity which affect the rapid innovation of C-ITS application development. In addition, with the booming of open data and data economy, the distributed nature of the traffic entities participating the network and the complex data privacy rules to protect traffic participants will shortly exceed the capacity of any traditional centralized systems. Under such a context, an innovation platform is needed where distributed and heterogeneous services can be coordinated and reused for fast C-ITS application prototyping. This is also driven by the emerging IoT era where each participant device and vehicle within the transport systems will be a service provider and can contribute to innovative traffic solutions.

This paper introduces the CHOReVOLUTION IDRE and describes how it helps to accelerate the development of a reuse-based choreography-based C-ITS application within the transport sector. It takes the UTC use case eco-driving as an example and showcases how such a platform can help innovators with rapid prototyping and validating their ideas, thus accelerating the innovation processes. The paper is organized as follows. Section II presents C-ITS from a system perspective and motivate the needs of a rapid innovation platform for the transport sector. Section III briefly describes the CHOReVOLUTION IDRE. Section IV introduces choreography diagrams and the related notation. Section V describes a UTC use case that provides eco-driving services for drivers to minimize the emission. Section VI presents how the CHOReVOLUTION IDRE has helped the rapid development of the UTC application in comparison with traditional development methods. Section VII analyzes the threats to validity. Section VIII discusses related works, and Section IX concludes the paper with discussion on future work.

II. C-ITS: System of Systems

A system of systems (SoS) is a set or an arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities [4]. C-ITS have the key characteristics of a SoS [5], namely autonomy, belonging, connectivity and diversity, and are thus a SoS. In C-ITS, participating systems, aka constituent systems, such as road infrastructure, communication services, vehicle systems, AI platforms, data provider systems are essentially independent systems with their own purposes of operation and evolution. They have different owners and are connected in a highly dynamic and distributed environment. The services provided by those systems are highly heterogeneous due to the complexity of C-ITS and the social-technical nature, and they collaborate in one way or another for a common purpose, such as to enable safe traffic and to reduce congestion and emissions. For example, vehicles can communicate and collaborate with each other voluntarily or through road infrastructure at intersections [6], [7] for safe, smooth and congestion-free passage. They can also form a platoon where vehicles are linked in convoy through connectivity with close distances between each other to enable safe and energy effective driving [8]. Connected emergency vehicles are able to communicate with other road vehicles so that they can give way in advance; and with road infrastructure so that it can empty the requested roads through advanced traffic control [9]. Emergent behaviors are the key characteristics of a SoS, where the effects of a SoS as a whole go beyond the cumulative effects of the individual systems. Such emergent behavior comes from the interactions and cooperation among all participating systems and is one of the key research focus areas for a SoS. For example, traffic jams are typical emergent behaviors since a single vehicle will not cause traffic jams. Such emergent behaviors are considered negative, but they exist due to the complex interactions among vehicles. On the other hand, proper cooperation between vehicles through C-ITS may support positive emergent behaviors that are preferred such as the above-mentioned collaborative applications.

A SoS can be classified into four types [10], namely directed, acknowledged, collaborative and virtual. A directed SoS is centrally managed where constituent systems are chosen and integrated for a specific purpose. In an acknowledged SoS, common purposes are recognized and a designated manager is in place. However, constituent systems are running independently and the common purposes are achieved through collaboration between a SoS and the constituent systems. This differs from a direct SoS in that the designated manager has no direct authority but to collaborate with each constituent system. For a collaborative SoS, constituent systems are more or less taking a voluntary position in participating the SoS, while for a virtual SoS, there is no recognized purpose or management authority.

C-ITS rely heavily on the interconnection and cooperation among constituent systems for a common goal that goes beyond the capability of a single participant system and a simple addition of the capabilities of all the participating systems. Considering the large number of heterogeneous services in the transport sector and the need for rapid innovation, C-ITS services will mostly involve acknowledged and collaborative SoSs. Engineering such a SoS goes beyond the capabilities of traditional system engineering and requires new tools and methods such as SoS engineering (SoSE) [11] to understand its SoS nature [12], [13].

In the meanwhile, service oriented approaches become popular methods for software development with distributed and loosely coupled systems. They leverage standardized and well-defined interfaces for system integration and promote interoperability, platform independence and the consideration of legacy systems. With service oriented architecture (SOA) [14], service composition can be centralized through orchestration or decentralized through choreography. While orchestration needs a single controller for coordinating the service interaction, choreography distributes the control logic among the participating services. The different composition methods associate with the different types of a SoS. In the case of C-ITS where acknowledged and collaborative SoS are anticipated, choreography is a promising service composition method for connecting and harmonizing a large (and uncertain) amount of heterogeneous service entities to build up an evolving base for rapid C-ITS application prototyping.

III. CHOReVOLUTION IDRE

This section briefly presents the CHOReVOLUTION IDRE. The CHOReVOLUTION IDRE, starting from a choreography specification and a set of existing services to be reused, solves the automatic realizability enforcement problem by automatically generating additional software artifacts to coordinate, adapt and secure the interactions among the services [15]. As shown in Figure 1, the CHOReVOLUTION IDRE consists of three layers: (1) a front-end layer; (2) a back-end layer; and (3) a flexible cloud infrastructure layer. Moreover, the front-end layer comprises (a) the CHOReVOLUTION Studio that is a design time Eclipse-based application; and (b) the CHOReVOLUTION console that is a web-based management and monitoring application. The full description of the CHOReVOLUTION IDRE can be found at.[1]

The IDRE mainly targets three types of users depicted in Figure 3 and described as follows.

Service providers interact with the CHOReVOLUTION Studio to publish existing services/things into the Service Inventory by defining their description models, i.e., interface and security models. In this way they encourage and facilitate the reuse of their services by developers, possibly leading to new business opportunities.

Choreography developers interact with the CHOReVOLUTION Studio to (i) specify a choreography by means of the Choreography Modeler. (ii) Realize the related choreography-based system by exploiting the Synthesis Processor to automatically synthesize Binding Components (for solving heterogeneity issues) [16], Coordination Delegates (for solving coordination issues) [17]–[22], Adapters (for solving interface

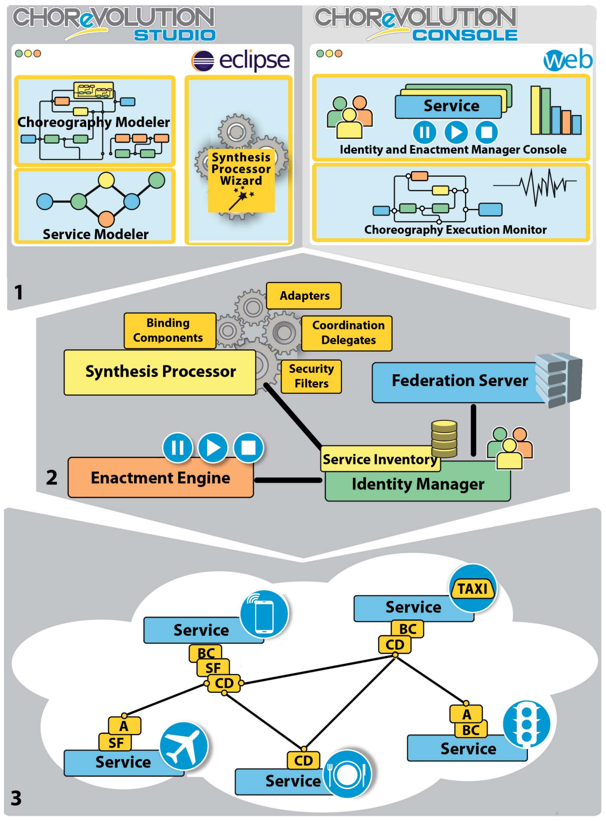


Fig. 1. CHOReVOLUTION IDRE overview.

mismatches) [23]–[25], and Security Filters (to make the choreography secure). The Coordination delegates are implemented as BPEL processes, whereas all the other artifacts are implemented as Java web services using the frameworks Apache CXF and Spring.

Choreography operators interact with the CHOReVOLUTION console to (i) deploy and enact the generated choreography-based system in the cloud; (ii) monitor the status of the cloud; (iii) monitor the execution and manage the lifecycle of the choreography instances.

IV. Choreography Diagrams

Choreography diagrams outline how business participants coordinate their interactions. The OMG's BPMN 2.0[2] standard introduced a dedicated notation for Choreography Diagrams that is the de facto standard for specifying choreographies. A BPMN2 choreography diagram models a distributed process defining the flows of the activities among the participants. In the following, the BPMN2 elements used in the UTC use case are described together with their semantic. An activity in a BPMN2 choreography diagram consists of a message exchange between two participants and it is represented by a choreography task (Figure 2 **(a)**). A choreography task is an atomic activity that represents an interaction by means of one or two message exchanges between two participants. In particular, the white box denotes the initiating participant (P1) that sends the initiating message (M1) to the

[1]http://www.chorevolution.eu/bin/view/Documentation/WebHome

[2]http://www.omg.org/spec/BPMN/2.0.2/

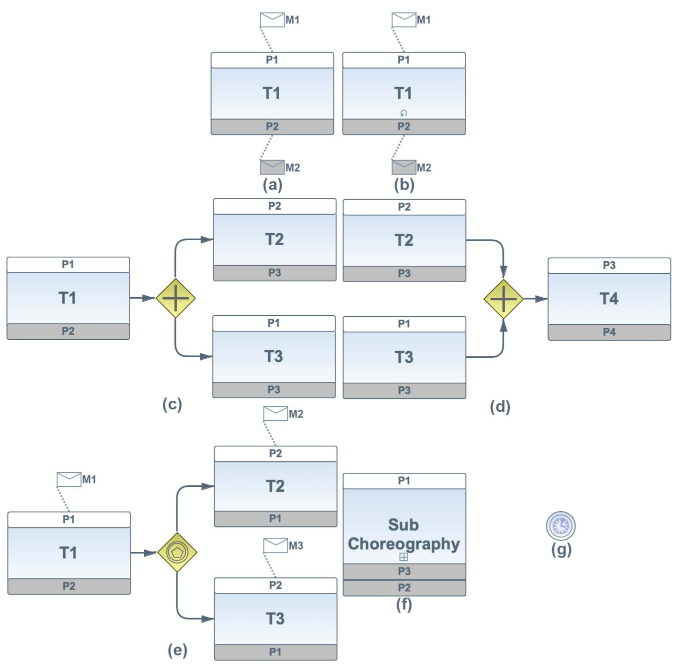


Fig. 2. BPMN2 choreography elements.

receiving participant in the dark grey band (P2). The receiving participant can optionally send back the return message (M2). A task can be looped (Figure 2 **(b)**) and hence it is performed a number of times depending on the evaluation of a specified conditional expression. A Parallel Gateway (Figure 2 **(c) & (d)**) is used to create ((c), i.e., diverging parallel gateway) and/or synchronize ((d), i.e., converging parallel gateway) parallel flows without checking any condition. For incoming flows, this gateway will wait for all incoming flows before triggering the flow through its outgoing arrow. They create parallel paths of the choreography known to all Participants. An Event-Based Gateway (Figure 2 **(e)**) is used to create alternative paths within a choreography, where the alternatives are based on events that occur at a certain point. This gateway is used when the data needed to make the decision is only internally visible to one participant. This means that there has been no message sent within the choreography that would expose the data used to make the decision. Therefore, the only way the other participants can be aware of the results of the decision is by the specific message that arrives next. A Sub-Choreography (Figure 2 **(f)**) is a composite activity that defines a flow of other tasks. Each sub-choreography involves two or more participants. In other words, a sub-choreography is a construct that allows for modular specification of choreographies. A timer event (Figure 2 **(g)**) is used to model a delay within the choreography flow. The delay mechanism can be based on a specific time-date or a specific time cycle.

V. Urban Traffic Coordination Use Case

The UTC use case targets eco-driving and, by means of a smartphone application, delivers real-time navigation services to advise the driver for proper actions aimed at reducing emissions. Such an application involves multiple stakeholders including the end users such as drivers and fleet owners, the data providers including commercial map and navigation suppliers, traffic information suppliers, weather information suppliers, etc., as well as development technologies such as the programming, integration and hosting platforms. In a previous work [26], the UTC use case has been presented with initial choreographies and experiments were conducted using an earlier version of CHOReVOLUTION platform without the IDRE. Since then, the platform has been significantly improved with the IDRE to automate the development process, which forms a major novelty of this paper. In addition, UTC choreographies have been improved and revised based on the lessons learned and have been implemented with the newest CHOReVOLUTION IDRE.

This section presents the latest UTC use case realized for the city of Gothenburg in collaboration with the related industrial partner[3] of the CHOReVOLUTION project. First, it describes the external provider services involved in the use case. Then, it describes the prosumer services that interact with the external services or with the other prosumer services to perform the UTC business logic and coordinate their interactions as designed by the choreography. Next, it presents the UTC choreography diagrams, and finally it briefly describes the SEADA client application.

A. UTC Provider Services

The provider services of the UTC use case provide routing information and different types of traffic information, such as congestion and accidents.

- **DTS-GOOGLE**: is the commercial routing services provided by Google through the Google Maps application programming interface (API). By requesting the route information with an origin and a destination, DTS-GOOGLE returns detailed descriptions of routing information.
- **DTS-HERE**: is an alternative routing services provided by HERE. It functions similarly to DTS-GOOGLE.
- **DTS-AREA-TRAFFIC**: is a provider service developed for area traffic information collection. It collects traffic information and it realizes a caching mechanism to provide real-time route traffic information. In fact, it receives all information within the area of interest and provides the most up-to-date route traffic information upon request. It has multiple interfaces serving different interaction purposes. One interface provides area segment information to, e.g., SEADA-TRAFFIC.
- **DTS-ACCIDENTS**: is a service that reports accidents information on the requested road segment. It is based on the real-time traffic information services provided by the Swedish Road Administration.
- **DTS-BRIDGE**: is a service that reports the bridge open status in the city of Gothenburg. It is provided by the city of Gothenburg through open data.
- **DTS-CONGESTION**: is a service that reports the congestion status for the requested road segment. It is currently provided by the Swedish Road Administration,

[3]https://www.ri.se/

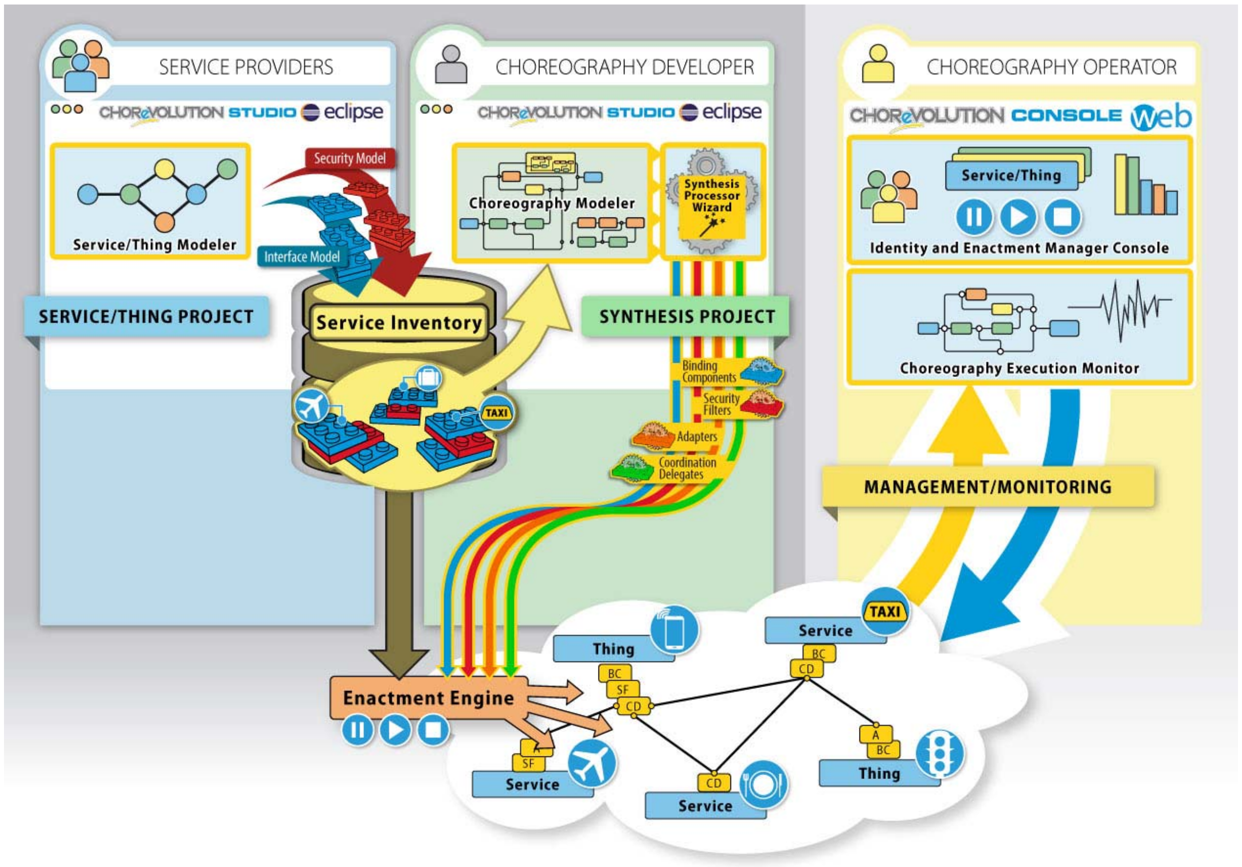


Fig. 3. CHOReVOLUTION IDRE users and main flows.

while many more commercial services can be readily used.

- **DTS-WEATHER**: is a service that reports the weather information on a requested road segment. The service is provided by the Swedish Meteorological and Hydrological Institute (SMHI).

B. UTC Prosumers

- **SEADA-SEATSA**: is the prosumer service that, as a provider, provides eco-evaluation for any given route. For evaluation and as a consumer service, it requests multiple traffic information for the given routes. The eco-evaluation is calculated by the internal logic of the service for CO2 emissions based on fuel consumption [27], which again depends on the road geometry, vehicle speed and acceleration. For detailed calculations the interested reader is referred to e.g., [28], [29].
- **SEADA-SEARP**: is the prosumer service that receives requests from end consumers for eco-route (provider side); and sends requests for route and traffic information required to suitably prepare eco-routes (consumer side). The routes are provided by DTS-GOOGLE and DTS-HERE, and eco-routes information is evaluated by the SEADA-SEATSA.
- **DTS-SEGMENT-TRAFFIC**: is responsible for interacting with all external services for traffic information and provides them for usage by other services within the choreography such as SEADA-TRAFFIC as discussed below.
- **SEADA-TRAFFIC**: is responsible for triggering the collection of traffic information for a certain area and storing the information locally. As a provider service, it provides all the traffic information, and as a consumer service, it requests all road segments information within an area of interest and collects traffic information for the road segments through DTS-SEGMENT-TRAFFIC.

C. UTC Choreographies

Two choreographies, namely SEADA and Traffic Information Collection have been designed in the UTC use case. The SEADA choreography contains all the coordination logic for eco-driving during the trip, where a sub-choreography is designed, namely Eco Friendly Routes Information sub-choreography, for eco-route evaluation. The Traffic Information Collection choreography is designed to focus on the traffic information collection and aggregation. A sub-choreography Traffic Segment Information Collection is designed for traffic information collection for a given road segment. The following part describes in detail the choreographies and their coordination logic.

1) SEADA Choreography: Figure 4 shows the SEADA choreography. The application is triggered when the driver starts the navigation app on the navigation device (ND) and

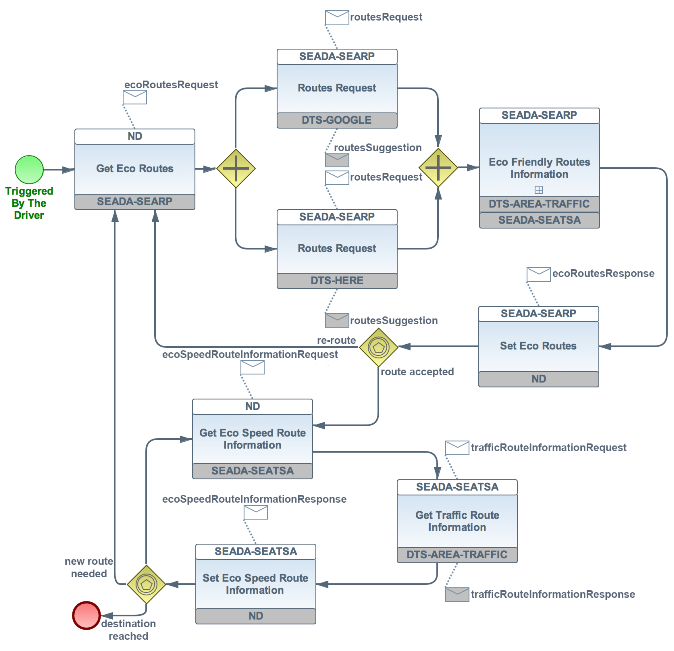


Fig. 4. SEADA choreography.

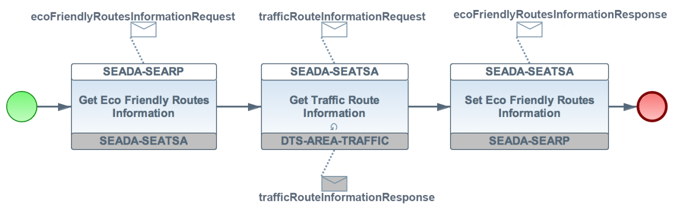

ecoFriendlyRoutesInformationRequest
trafficRouteInformationRequest
ecoFriendlyRoutesInformationResponse
SEADA-SEARP
Get Eco Friendly Routes Information
SEADA-SEATSA
SEADA-SEATSA
Get Traffic Route Information
DTS-AREA-TRAFFIC
SEADA-SEATSA
Set Eco Friendly Routes Information
SEADA-SEARP
trafficRouteInformationResponse

Fig. 5. Eco friendly routes information sub-choreography.

inputs the origin and destination information. Next, ND interacts with SEADA-SEARP through the *Get Eco Routes* task. After, SEADA-SEARP simultaneously retrieves routes information from DTS-GOOGLE and DTS-HERE. These routes information are then integrated by the related eco-friendly information and returned to ND that chooses an eco-optimized route. While driving according to the given route, ND continuously updates traffic information with the purposes of giving driving advisory to the drivers for eco-friendly driving. Indeed, it interacts with SEADA-SEATSA that collects traffic information from DTS-AREA-TRAFFIC, evaluates the eco factors and then gives back eco-speed information.

Figure 5 shows the Eco Friendly Routes Information sub-choreography. It consists of SEADA-SEARP requesting eco information for the available routes to SEADA-SEATSA that in turn interacts with DTS-AREA-TRAFFIC to get information for each route. This is done through the looped task Get Traffic Route Information. After all information is available, SEADA-SEATSA returns the routes with eco information to SEADA-SEARP.

2) Traffic Information Collection Choreography: As shown in Figure 6, Traffic information collection choreography is a long-running choreography. In fact, it is designed to continuously collect traffic related information that are cached by DTS-AREA-TRAFFIC. Leveraging the city area related information managed by Traffic Information Collector,

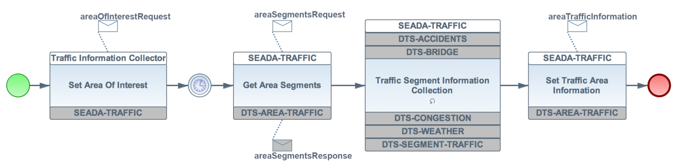


Fig. 6. Traffic information collection choreography.

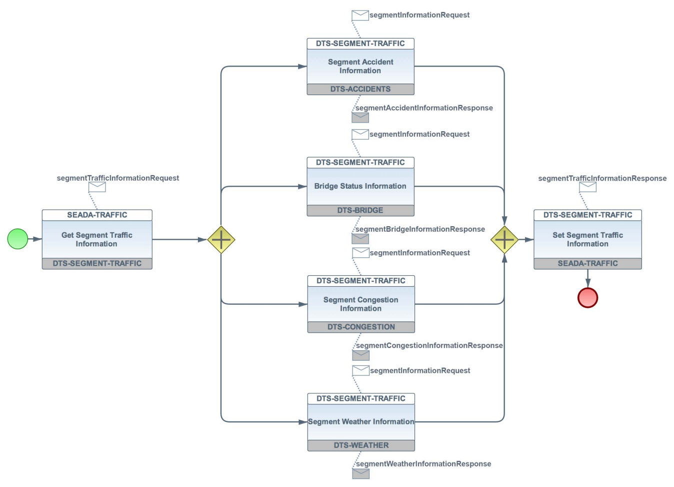

segmentInformationRequest
DTS-SEGMENT-TRAFFIC
Segment Accident Information
DTS-ACCIDENTS
segmentAccidentInformationResponse
segmentInformationRequest
DTS-SEGMENT-TRAFFIC
Bridge Status Information
DTS-BRIDGE
segmentBridgeInformationResponse
segmentTrafficInformationRequest
SEADA-TRAFFIC
Get Segment Traffic Information
DTS-SEGMENT-TRAFFIC
segmentTrafficInformationResponse
DTS-SEGMENT-TRAFFIC
Set Segment Traffic Information
SEADA-TRAFFIC
segmentInformationRequest
DTS-SEGMENT-TRAFFIC
Segment Congestion Information
DTS-CONGESTION
segmentCongestionInformationResponse
segmentInformationRequest
DTS-SEGMENT-TRAFFIC
Segment Weather Information
DTS-WEATHER
segmentWeatherInformationResponse

Fig. 7. Traffic segment information collection sub-choreography.

SEADA-TRAFFIC periodically requests segments information for a given area, collects the traffic information for each segment through the looped execution of the sub-choreography Traffic Segment Information Collection and then return it to DTS-AREA-TRAFFIC. In this way DTS-AREA-TRAFFIC is able to provide the most up-to-date traffic information for the segments of a route upon request, as in the tasks Get Traffic Route Information of the previous choreographies.

Figure 7 illustrates the Traffic Segment Information Collection sub-choreography. SEADA-TRAFFIC interacts with DTS-SEGMENT-TRAFFIC for traffic information for a given segment. Then, DTS-SEGMENT-TRAFFIC concurrently gets accident information, bridge status information, congestion information and weather information for a segment. Finally, after collecting this information DTS-SEGMENT-TRAFFIC provides the segment traffic information to SEADA-TRAFFIC.

D. SEADA Client Application

To exploit the functionalities provided by the synthesized choreographies, a client that plays the role of ND in the SEADA choreography has been developed. Figure 8 illustrates the client interface on a smartphone. The client shows turn by turn navigation instructions including real-time traffic situations and eco related information, consisting of: current driving speed, suggested eco-driving speed, estimated CO2 emission in comparison with the suggested driving behavior, traffic condition of the next route segment, bridge (green marked) open/close status, and weather condition as provided by the SEADA choreography.

VI. EVALUATION

The realization of the UTC use case using the CHOReVOLUTION IDRE has been experimentally evaluated against a

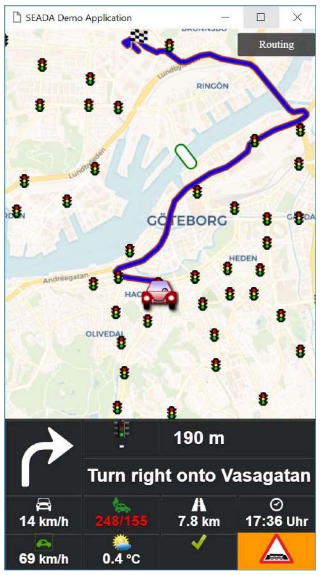


Fig. 8. SEADA client interface.

set of Key Performance Indicators (KPIs) to assess the resulting benefits. In particular, the experiment aims to measure the time saving for realizing and maintaining/evolving the UTC use case with the CHOReVOLUTION approach, when compared to the development approach the partners daily use.

The considered KPIs are:

- **Effort for designing the system** – The specification of the services involved in the system and their interaction flows;
- **Effort for implementing the distributed workflow (coordination logic)** – the distributed coordination logic guarantees the collaboration among the involved services;
- **Effort for implementing the missing logic (business logic)** – the missing business logic that must be coded from scratch;
- **Effort for integrating third-party or legacy services** – the integration code needed to reuse and adapt heterogeneous third-party or legacy services.

The KPIs are measured at three different software development phases: **implementation**, **maintenance** and **evolution**. The implementation phase consists of the development of a choreography-based system from scratch. The maintenance phase concerns the implementation of updates through service substitution. The evolution phase concerns the development effort required to tackle business goal changes through the modification of the choreography specification.

Each phase includes testing activities to assess the correctness of the system implementation with respect to the execution flows defined in the choreography specification. These activities have been performed according to the testing practices the industrial partner normally adopts. The testing activities concern specific development work of the industrial partner that is out of the scope of this paper. Strictly concerning the purpose of this paper, it is enough to note that the model-driven approach for the synthesis of the Coordination Delegates is correct by construction, as formally proved in [20]. The approach implementation within the CHOReVOLUTION IDRE and the artifacts it generates have been validated through several test cases and use cases beyond the UTC use case presented in this paper. In particular, regarding the UTC use case, the test cases did not discover any errors in the artifacts generated by the CHOReVOLUTION approach; whereas, several errors were discovered in the artifacts coded by the industrial partner, especially for what concerns the distributed coordination logic.

Back to the considered phases, the experiment aims to test the following hypotheses: the CHOReVOLUTION approach allows developers to implement (**Hypothesis 1**), maintain (**Hypothesis 2**), and evolve (**Hypothesis 3**) a choreography-based system more quickly. The time saving is measured in terms of person-hour (ph). In particular, concerning the UTC use case, we employed the following experimental units.

Experimental unit 1 – *CHOReVOLUTION approach*: full usage of the CHOReVOLUTION IDRE.

Experimental unit 2 – *General-purpose enterprise-oriented technology*: full usage of the technologies daily adopted by the industrial partner, i.e., NodeJS, ExpressJS framework together with other supporting libraries (xml2js, node-soap, nodemon), and Visual Studio.

The technologies of the experimental unit 2 were selected considering that the industrial partner was familiar and skilled with them. It is clear that there exist many other equivalently powerful alternatives. However, opting for an alternative would have required a training effort that could not have been afforded by the partner because of budget constraints. In any case, apart from budget constraints and assuming the possibility to opt for an alternative, we can argue that, whatever (reasonably) long the training could last, it would not have been easy to reach the same level of expertise, hence possibly compromising the validity of the experiment.

By referring to Table I, the considered KPIs are mapped into experimental tasks. In particular, the effort for designing the system is mapped into the system design; the effort for implementing the distributed workflow is mapped into the coordination logic implementation; the effort for coding the missing logic is mapped into the prosumer services implementation; finally, the effort for reusing third-party or legacy services is mapped into the adaptation logic implementation. Note that the system design is performed only during the implementation phase, whereas all the other tasks are performed in each phase of the experiment.

The experiment was conducted by four developers, namely Dev1, Dev2, Dev3 and Dev4. Dev1 and Dev2 were engaged in the use case development for the CHOReVOLUTION project, whereas Dev3 and Dev4 came from a different project. As reported in Table II, when assigning the experimental tasks to the developers: Dev1, Dev2 and Dev3, we took care to eliminate the potential bias of person-task links. This means that none of them was involved in the same task across the two experimental units. Instead, Dev4 was involved in the system design task in both experimental units. In fact, Dev4 is an expert researcher and a domain expert that supervises the design task to ensure that the two systems specified in the different notations are the same in terms of the admissible flows among the business tasks, and hence they require the same

TABLE I
UTC EXPERIMENTAL TASKS ACTIVITIES

Experimental tasks	**Experimental unit 1**	**Experimental unit 2**
System design	The services involved in the system and their interaction flows are specified in terms of BPMN2 choreography diagrams	The services involved in the system and their interaction flows are specified by means of the informal box-line notation adopted by the industrial partner
Coordination logic	The code realizing the distributed coordination logic is automatically generated into a set of Coordination Delegates, without requiring any manual intervention	The coordination logic and the business logic for each of the distributed workflows is implemented with NodeJS and Visual Studio Code IDE. In particular, the developers have to design the session handling mechanism and the mechanism to realize the distributed workflows (sequences of tasks, parallel and event-based gateways).
Prosumer services	The skeleton code of the prosumer services is automatically generated. Thus, developers are required to only fill in the blanks of highlighted and partially ready pieces of code	The prosumer services are manually implemented. In particular, for each choreography task involving a specific participant, the logic to manipulate the received messages and to build the messages to be sent needs to be coded from scratch. The developers have to maintain the data and message storage to ensure that the messages are well parsed and routed through different distributed flows with no data loss.
Adaptation logic	The adaptation logic is automatically generated into a set of Adapters	The adaptation logic to bind the concrete services to the choreography participants in case of interface mismatches is manually implemented.

TABLE II
EXPERIMENTAL TASKS ASSIGNMENT

Experimental tasks	**Experimental unit 1**	**Experimental unit 2**
System design	Dev1, Dev4	Dev3, Dev4
Coordination logic	Dev1	Dev3
Prosumer services	Dev2	Dev1
Adaptation logic	Dev3	Dev2

TABLE III
EXPERIMENTAL TASKS RESULTS - IMPLEMENTATION PHASE

Tasks	**Experimental unit 1 (ph)**	**Experimental unit 2 (ph)**
System design	120 **(+ 12)**	80
Coordination logic	0	120
Prosumer services	5	14
Adaptation logic	3	18
Total	**140**	**232 (92)**

coordination logic. As better discussed in Section VII, Dev1, Dev2 and Dev3, although assigned to different experimental tasks, were employed in both experimental units due to budget and time constraints imposed by the CHOReVOLUTION project.

All developers had equivalent professional skills, and familiarity with the CHOReVOLUTION IDRE. In fact, they were trained in the use of CHOReVOLUTION IDRE through the following two dedicated workshops:

- 1st CHOReVOLUTION Hands-on Workshop on Advanced Service-oriented Software Engineering, "CHOReVOLUTION meets Master's degree Students", May 8-9, 2017, University of L'Aquila, L'Aquila, Italy.
- 2nd CHOReVOLUTION Workshop, "CHOReVOLUTION Hands-on Workshop", RISE Viktoria, November 16, 2017, Lindholmspiren, Goteborg, Sweden.

The training required 12 ph, resulting in a negligible learning curve if compared with the usual time needed by the employees of a company to learn a new technology.

Hypothesis 1: We found that the proposed approach significantly decreased the time required to implement the use case. Table I describes the activities performed within each experimental unit for accomplishing the experimental tasks. It is worth noticing that, regarding the tasks: coordination logic, prosumer services and adaptation logic, the proposed approach provides a higher support to automation with respect to the other approaches, which require a manual implementation.

For each experimental unit, Table III reports the ph employed to carry out the experimental tasks together with the total amounts of ph. It is worth to note that, concerning the coordination logic implementation task in the experimental unit 1, a developer has only to select the possible correlated choreography tasks in the wizard of the CHOReVOLUTION IDRE. Hence, we considered the required time as 0 ph. Regarding the system design task in the experimental unit 1, the total amount of ph employed in the task for the experimental unit 1 denotes in brackets the time required to train the developers in the use of the CHOReVOLUTION IDRE. Whereas, the total amount for the experimental units 2 highlights in brackets the ph saved by using the proposed approach. Although the CHOReVOLUTION approach performed worse in the system design task, on the whole, it performed better. In fact, the general-purpose enterprise-oriented approach took more than 0.6 times longer than the CHOReVOLUTION approach.

TABLE IV
EXPERIMENTAL TASKS RESULTS - MAINTENANCE PHASE

Tasks	**Experimental unit 1 (ph)**	**Experimental unit 2 (ph)**
Coordination logic	0	8
Prosumer services	0,2	1,5
Adaptation logic	0,5	3
Total	**0,7**	**12,5 (11,8)**

TABLE V
EXPERIMENTAL TASKS RESULTS - EVOLUTION PHASE

Tasks	**Experimental unit 1 (ph)**	**Experimental unit 2 (ph)**
Coordination logic	0	20
Prosumer services	1	3,5
Adaptation logic	0,5	3
Total	**1,5**	**26,5 (25)**

Hypothesis 2: We found that the proposed approach provides a meaningful decrease of the time required to maintain the UTC choreography-based system. In the maintenance phase, a different service is selected to play the role of the participant `DTS-HERE`. This results in a service substitution. The selected service has a different interface with respect to the one required by the choreography specifications, hence, an adapter is needed. In particular, the proposed approach provides automatic support to the generation of adapters, whereas the other approach requires a manual implementation.

For each experimental unit, Table IV reports the ph employed to carry out the experimental tasks together with the total amounts of ph. The total amount for the experimental unit 2 highlights in brackets the ph saved by using the proposed approach. The general-purpose enterprise-oriented approach took more than sixteen times longer than the proposed approach.

Hypothesis 3: We found that the proposed approach significantly reduces the time required to evolve the UTC choreography-based system. In the evolution phase, the `Traffic Segment Information Collection` sub-choreography is modified by introducing the choreography task `Segment Traffic Light Information` between the participant `DTS-SEGMENT-TRAFFIC` and the new participant `DTS-TRAFFICLIGHT`. The service selected to play the role of the participant `DTS-TRAFFICLIGHT` has a different interface with respect to the one required by the choreography specification. Thus, an adapter, a new coordination delegate and a new prosumer corresponding to the `DTS-SEGMENT-TRAFFIC` participant are needed.

For each experimental unit, Table V reports the ph employed to carry out the experimental tasks of the UTC use case together with the total amounts of ph. The general-purpose enterprise-oriented approach took more than sixteen times longer than the proposed approach. In this phase the UTC choreography has been modified by adding a new choreography task together with a new participant. This allowed all the experimental units to leverage on code reuse. It is worth noting that the time saving obtained in this phase is due to the high support to automation provided by the proposed approach with respect to the other approach that requires a manual implementation although reusing some code.

TABLE VI
OVERALL CALCULATION OF TIME SAVING

Experimental units	**Implementation (ph)**	**Maintenance (ph)**	**Evolution (ph)**	**Time saving (ph)**
1	140	0,7	1,5	**–**
2	232 **(92)**	12,5 **(11,8)**	26,5 **(25)**	**128,8**

Experiment Results

Table VI summarizes the results of the experiment on the UTC use case by distinguishing the implementation, maintenance, and evolution phases. In particular, the experimental unit 2 highlights in bold the person-hours saved by using the proposed approach. It is worth noting that the proposed approach results in a decrease of the required development time in all the considered development phases: implementation (hypothesis 1), maintenance (hypothesis 2) and evolution (hypothesis 3). Concerning the implementation phase, the decrease of the development time is due to the high support for automation provided by the proposed approach with respect to the other approach. Regarding the maintenance and the evolution phases, the decrease is more significant in the latter, where the changes affect the choreography specification, than in the maintenance phase, where the changes affected the services involved in the choreography-based system. Moreover, the last column contains the total amounts of ph saved. This result together with the amount of ph saved reveals that the proposed approach has great potential in developing choreography-based systems and the use case got a full benefit from it.

As reported in https://github.com/chorevolution/CHOReVOLUTION-IDRE/wiki/Experiments, by considering the feedback from the industrial partners, we can state that the advantages showed in UTC use case consist in: reusing existing services, support for distributed composition, support for automation of coding operations and provision of correctness by construction. These factors can effectively contribute to the success of CHOReVOLUTION IDRE to create business opportunities. The most evident scenarios to which the outcomes of the experiment can be applicable are content integration, distributed workflow and business process management, especially in the Smart Cities context. An example are Mobility-as-a-Service applications in which stakeholders, public and private ITS organizations (public transport companies, parking companies, etc.) cooperate to offer static and dynamic information. In general, the benefits offered by the programming paradigm of the CHOReVOLUTION IDRE to ICT companies are: software reuse, rapid prototyping and agile development.

VII. Threats to Validity

The experiment reported in this paper aimed at comparing the CHOReVOLUTION approach with the development approach daily used by the industrial partner. This experiment was required by the EU commission. Ideally, the experiment should have involved the CHOReVOLUTION IDRE and all the existing tools for choreography specification and choreography system development, as the ones reported in the related work section. However, this was unfeasible within the context of the CHOReVOLUTION project due to budget, time and availability constraints. Moreover, it would have required a deep study of each framework to find a common, yet best fitting, comparison strategy and hence it would have deserved a dedicated paper. In fact, as explained in Section VIII, although the other tools apparently address similar foundational issues, in the practice of distributed system development there are several substantial differences.

Concerning the developers involved in the experiment, although assigned to different experimental tasks, they were involved in both the first and the second experimental unit due to developers availability, budget and time constraints of the CHOReVOLUTION project. However, leveraging on the expertise of the industrial partner, we made an estimate of the time needed in the experimental unit 2 without developers already familiar with the UTC use case and we can conclude that it would have been higher, hence representing a further advantage towards our approach.

VIII. Related Work

VerChor is a framework for choreography design and verification [30]. The framework translates the existing choreography languages, i.e., conversation protocols or BPMN 2.0 choreographies, into the LOTOS NT (LNT) process algebra [31] by means of a choreography intermediate format (CIF). In this way, the framework exploits the analysis techniques provided by the CADP [32] toolbox to check a set of key properties that choreographies must respect to ensure correctness of the system under development. Our approach leverages on theoretical results exploited in [30] to provide practical and automatic support for the realization of reuse-based service choreographies and hence to boost the adoption of choreographies in the development practices adopted by IT companies.

The ASTRO toolset supports automated composition of services [33]. Its purpose is the composition of a service out of a business requirement and the description of available external services. More specifically, a planner component automatically synthesizes the code of a centralized process that achieves the business requirement by interacting with the available external services. Differently from our approach, ASTRO handles orchestration-based processes rather than decentralized choreography-based ones.

The CIGAR (Concurrent and Interleaving Goal and Activity Recognition) framework aims for multigoal recognition [34]. CIGAR decomposes an observed sequence of multigoal activities into a set of action sequences, one for each goal, specifying whether a goal is active in a specific action. Although such goal decomposition somehow recalls our choreography decentralization, goal recognition represents a fundamentally different problem compared to realizability enforcement. Goal recognition concerns learning a goal-based model of an agent by observing the agent's actions while interacting with the environment. In contrast, realizability enforcement produces a decentralised coordination logic out of a choreography specification.

Carbone *et al.* [35] present a unified programming framework for developing choreographies that are correct by construction in the sense that, e.g., they ensure deadlock freedom and communication safety. Developers can design both protocols and implementation from a global perspective and, then, correct endpoint implementations are automatically generated. The work in [35] considers the notion of multiparty choreography and defines choreography projection in a way that the endpoint code is correctly generated from the designed choreography. In [36], the authors discuss some of the extensions of the Jolie orchestration language [37] developed in the context of the EU Sensoria project.[4] One of the extensions concerns the JoRBA framework that allows to develop dynamically adaptable service oriented applications. A further extension regards the Chor language that allows for the programming of distributed systems from a global perspective. Chor is equipped with code generation tools that produce correct-by-construction Jolie programs, with respect to the originally designed system. The last extension is the AIOCJ choreography language that, leveraging the adaptability features of JoRBA, allows for developing adaptable choreographies. Differently from our approach, the focus of the works described in [35], [36] is more on developing choreographies from scratch, rather than realizing them through the reuse of existing, often black-box, participants.

IX. Conclusion and Future Work

This paper reported on the exploitation of the CHOReVOLUTION IDRE to accelerate the reuse-based development of a choreography-based C-ITS. In particular, it described the realization of a UTC use case in the city of Gothenburg to provide eco-driving services for drivers to minimize the emissions. The eco-evaluation is an initial implementation and more advanced methods are expected to be included in the future. In particular, a more accurate prediction can be provided by considering additional traffic information and employing machine learning methods. Another future work is the integration of vehicle on-board systems to communicate directly with vehicle control systems.

References

[1] H. Ritchie. (2018). Urbanization. Our World Data. [Online]. Available: https://ourworldindata.org/urbanization

[2] *Greenhouse Gas Emissions From Transport in Europe*, Eur. Environ. Agency, Copenhagen, Denmark, 2019.

[4]http://www.sensoria-ist.eu

[3] L. Chen and C. Englund, "Cooperative ITS—EU standards to accelerate cooperative mobility," in *Proc. Int. Conf. Connected Vehicles Expo (ICCVE)*, Nov. 2014, pp. 681–686.

[4] *Systems Engineering Guide for Systems of Systems*, USDOD, Richmond, VA, USA, Aug. 2008.

[5] J. Boardman and B. Sauser, "System of systems—The meaning of of," in *Proc. IEEE/SMC Int. Conf. Syst. Syst. Eng.*, Apr. 2006, pp. 118–123.

[6] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, Feb. 2016.

[7] C. Englund *et al.*, "The grand cooperative driving challenge 2016: Boosting the introduction of cooperative automated vehicles," *IEEE Wireless Commun.*, vol. 23, no. 4, pp. 146–152, Aug. 2016.

[8] H. H. Bengtsson, L. Chen, A. Voronov, and C. Englund, "Interaction protocol for highway platoon merge," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst.*, Sep. 2015, pp. 1971–1976.

[9] L. Chen and C. Englund, "Every second counts: Integrating edge computing and service oriented architecture for automatic emergency management," *J. Adv. Transp.*, vol. 2018, Feb. 2018, Art. no. 7592926. [Online]. Available: https://www.hindawi.com/journals/jat/2018/7592926/

[10] M. W. Maier, "Architecting principles for systems-of-systems," *Syst. Eng.*, vol. 1, no. 4, pp. 267–284, 1998.

[11] C. B. Keating and P. F. Katina, "Systems of systems engineering: Prospects and challenges for the emerging field," *Int. J. Syst. Syst. Eng.*, vol. 2, nos. 2–3, p. 234, 2011.

[12] D. A. DeLaurentis, "Understanding transportation as a system of systems problem," in *System of Systems Engineering*. Hoboken, NJ, USA: Wiley, 2008, ch. 20, pp. 520–541. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470403501.ch20, doi: 10.1002/9780470403501.ch20.

[13] P. G. Teixeira *et al.*, "Constituent system design: A software architecture approach," in *Proc. IEEE Int. Conf. Softw. Archit. Companion (ICSA-C)*, Mar. 2020, pp. 218–225.

[14] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, "Reference model for service oriented architecture 1.0, committee specification 1," OASIS, Adobe Syst. Incorp., San Jose, CA, USA, MITRE Corp., Bedford, MA, USA, Fujitsu Ltd., New York, NY, USA, Independ. Consultant, Los Angeles, CA, USA, Booz Allen Hamilton, McLean, VA, USA, Tech. Rep. 1.0, Aug. 2006.

[15] M. Autili, A. D. Salle, F. Gallo, C. Pompilio, and M. Tivoli, "CHOReVOLUTION: Service choreography in practice," *Sci. Comput. Program.*, vol. 197, Oct. 2020, Art. no. 102498.

[16] G. Bouloukakis, "Enabling emergent mobile systems in the IoT: From middleware-layer communication interoperability to associated QoS analysis," Ph.D. dissertation, Inria, Paris, France, 2017.

[17] M. Autili, A. D. Salle, F. Gallo, C. Pompilio, and M. Tivoli, "Aiding the realization of service-oriented distributed systems," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, Limassol, Cyprus, Apr. 2019, pp. 1701–1710.

[18] M. Autili, D. D. Ruscio, A. D. Salle, P. Inverardi, and M. Tivoli, "A model-based synthesis process for choreography realizability enforcement," in *Proc. 16th Int. Conf. Fundam. Approaches Softw. Eng. (FASE)*, Rome, Italy, Mar. 2013, pp. 37–52.

[19] M. Autili, D. D. Ruscio, A. D. Salle, and A. Perucci, "CHOReOSynt: Enforcing choreography realizability in the future Internet," in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng. (FSE)*, Hong Kong, Nov. 2014, pp. 723–726.

[20] M. Autili, P. Inverardi, and M. Tivoli, "Choreography realizability enforcement through the automatic synthesis of distributed coordination delegates," *Sci. Comput. Program.*, vol. 160, pp. 3–29, Aug. 2018.

[21] M. Autili, A. D. Salle, F. Gallo, C. Pompilio, and M. Tivoli, "On the model-driven synthesis of evolvable service choreographies," in *Proc. 12th Eur. Conf. Softw. Archit., Companion (ECSA)*, Sep. 2018, pp. 20:1–20:6.

[22] M. Autili, P. Inverardi, A. Perucci, and M. Tivoli, "Synthesis of distributed and adaptable coordinators to enable choreography evolution," in *Software Engineering for Self-Adaptive Systems III. Assurances—International Seminar, Dagstuhl Castle, Germany, December 15–19, 2013, Revised Selected and Invited Papers* (Lecture Notes in Computer Science), vol. 9640, R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, Eds. New York, NY, USA: Springer, 2013, pp. 282–306. [Online]. Available: https://doi.org/10.1007/978-3-319-74183-3_10, doi: 10.1007/978-3-319-74183-3_10.

[23] M. Autili, A. D. Salle, F. Gallo, C. Pompilio, and M. Tivoli, "On the model-driven synthesis of adaptable choreographies," in *Proc. MODELS Workshops*, Copenhagen, Denmark, Oct. 2018, pp. 12–17.

[24] M. Autili, A. D. Salle, F. Gallo, C. Pompilio, and M. Tivoli, "Model-driven adaptation of service choreographies," in *Proc. 33rd Annu. ACM Symp. Appl. Comput. (SAC)*, Apr. 2018, pp. 1441–1450.

[25] A. D. Salle, F. Gallo, and A. Perucci, "Towards adapting choreography-based service compositions through enterprise integration patterns," in *Proc. SEFM Collocated Workshops*, York, U.K., Sep. 2015, pp. 240–252.

[26] L. Chen and C. Englund, "Choreographing services for smart cities: Smart traffic demonstration," in *Proc. IEEE 85th Veh. Technol. Conf. (VTC Spring)*, Jun. 2017, pp. 1–5.

[27] *Average Carbon Dioxide Emissions Resulting From Gasoline and Diesel Fuel*, United States Environ. Protection Agency, Washington, DC, USA, 2005.

[28] W. Zeng, T. Miwa, and T. Morikawa, "Prediction of vehicle CO_2 emission and its application to ECO-routing navigation," *Transp. Res. C, Emerg. Technol.*, vol. 68, pp. 194–214, Jul. 2016.

[29] L. Guzzella and A. Sciarretta, *Vehicle Propulsion Systems*. Berlin, Germany: Springer, 2013. [Online]. Available: https://doi.org/10.1007%2F978-3-642-35913-2, doi: 10.1007/978-3-642-35913-2.

[30] M. Gudemann, P. Poizat, G. Salaun, and L. Ye, "VerChor: A framework for the design and verification of choreographies," *IEEE Trans. Services Comput.*, vol. 9, no. 4, pp. 647–660, Jul. 2016.

[31] D. Champelovier *et al.*, "Reference manual of the LOTOS NT to LOTOS translator (version 5.4)," INRIA-VASY Res. Team, Montbonnot-Saint-Martin, France, Tech. Rep. 5.4, 2011, p. 149.

[32] H. Garavel, F. Lang, R. Mateescu, and W. Serwe, "CADP 2010: A toolbox for the construction and analysis of distributed processes," in *Proc. 17th Int. Conf. Tools Algorithms Construct. Anal. Syst.*, Saarbrücken, Germany, Mar. 2011, pp. 372–387.

[33] M. Trainotti *et al.*, "ASTRO: Supporting composition and execution of Web services," in *Proc. 3rd Int. Conf. Service-Oriented Comput.*, Amsterdam, The Netherlands, Dec. 2005, pp. 495–501.

[34] D. H. Hu and Q. Yang, "CIGAR: Concurrent and interleaving goal and activity recognition," in *Proc. 23rd AAAI Conf. Artif. Intell. (AAAI)*, Chicago, IL, USA, Jul. 2008, pp. 1363–1368.

[35] M. Carbone and F. Montesi, "Deadlock-freedom-by-design: Multiparty asynchronous global programming," in *Proc. 40th Symp. Princ. Program. Lang.*, 2013, pp. 263–274.

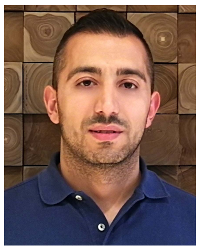
[36] I. Lanese, F. Montesi, and G. Zavattaro, "The evolution of jolie—From orchestrations to adaptable choreographies," in *Software, Services, and Systems—Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering* (Lecture Notes in Computer Science), vol. 8950, R. De Nicola and R. Hennicker, Eds. Springer, 2015, pp. 506–521. [Online]. Available: https://doi.org/10.1007/978-3-319-15545-6_29, doi: 10.1007/978-3-319-15545-6_29.

[37] F. Montesi, C. Guidi, R. Lucchi, and G. Zavattaro, "JOLIE: A java orchestration language interpreter engine," *Electron. Notes Theor. Comput. Sci.*, vol. 181, pp. 19–33, Jun. 2007.

Marco Autili received the Ph.D. degree in computer science from the University of L'Aquila in 2008. He is currently an Associate Professor with the University of L'Aquila. He is (has been) involved in several EU and Italian research and development projects, as a Scientific Coordinator, a Scientific and a Technical Leader, a Research Unit Coordinator, and a Work Package Leader. His research interests include automated synthesis for composing distributed systems, context-oriented privacy-aware mobile software programming, resource-oriented analysis of mobile apps, formal specification, and checking of temporal properties. He is on the Editorial Board and in the programme committee of several top-level international journals, international conferences and workshops. More information is available at https://people.disim.univaq.it/marco.autili/

Lei Chen received the Ph.D. degree in infra-informatics from Linköping University, Norrköping, Sweden. He is currently a Senior Researcher with the RISE Research Institute of Sweden. At RISE, he focuses on future mobility that is enabled by information and communication technologies (ICT). His research interests include mathematical optimization, mobile telecommunication, and connected automated vehicles and transport.

Cristofer Englund received the Ph.D. degree in electrical engineering from the Chalmers University of Technology, Gothenburg, Sweden, in 2007. He is currently a Research Director within humanized autonomy at RISE Viktoria, Gothenburg. He is also an Adjunct Professor with Halmstad University within machine learning. His research interests include autonomous driving, behavior modeling, and data mining.



Claudio Pompilio received the Ph.D. degree in computer science from the University of L'Aquila. He is currently a Post-Doctoral Researcher with the Department of Information Engineering, Computer Science, and Mathematics, University of L'Aquila. His research interest includes engineering of service-oriented distributed systems based on choreographies. In particular, his research is mainly focused on the automatic synthesis of components responsible for the coordination and adaptation aspects. The results of his research has been applied in the EU H2020 CHOReVOLUTION project. Contact him at claudio.pompilio@univaq.it.

Massimo Tivoli is currently an Associate Professor with the DISIM, University of L'Aquila. He was and is currently involved in several European and National projects, among which the European H2020 CHOReVOLUTION project, where he was Scientific Coordinator. His main research interest includes definition and application of formal software engineering methods to the development of distributed software systems. In particular, one of his research topics include the automated synthesis of correct-by-construction connectors for the correct assembly of distributed component-based systems, focusing on the production of both centralized and distributed connectors. Other research topics include the development of dependable and adaptable systems, automated methods to learn the interaction protocol performed by a service directly out of its signature description, and automated choreography synthesis approaches. He is a reviewer of several leading international peer-reviewed journals. He participated and participates to Program Committees of several relevant conferences on software architectures and software engineering. He has been a Program Chair of the international symposium CBSE2013. More information is available at http://people.disim.univaq.it/massimo.tivoli