Science of Computer Programming 197 (2020) 102498

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Original software publication

# CHOReVOLUTION: Service choreography in practice

Marco Autili, Amleto Di Salle[*], Francesco Gallo, Claudio Pompilio, Massimo Tivoli

*Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Via Vetoio snc, L'Aquila, Italy*

ARTICLE INFO

*Article history:*
Received 15 November 2019
Received in revised form 2 June 2020
Accepted 2 June 2020
Available online 8 June 2020

*Keywords:*
Service choreographies
Distributed computing
Distributed coordination
Automated synthesis

ABSTRACT

This paper presents CHOReVOLUTION, a platform for the tool-assisted realization and execution of distributed applications. CHOReVOLUTION specifically targets service-oriented systems specified through service choreographies. It offers an Integrated Development and Runtime Environment (IDRE) organized into three layers, namely, front-end, back-end, and cloud. It comprises a wizard-aided development environment and a system monitoring console in the front-end layer, and a back-end for managing the deployment and execution of the choreographed system on the cloud. We describe the IDRE by using an industrial use case in the domain of Smart Mobility & Tourism, and finally we provide details on its experimental evaluation.

* Corresponding author.
*E-mail addresses:* marco.autili@univaq.it (M. Autili), amleto.disalle@univaq.it (A. Di Salle), francesco.gallo@univaq.it (F. Gallo), claudio.pompilio@univaq.it (C. Pompilio), massimo.tivoli@univaq.it (M. Tivoli).

https://doi.org/10.1016/j.scico.2020.102498
0167-6423/© 2020 Elsevier B.V. All rights reserved.

**Table 1**
Software metadata (optional).

| (executable) Software metadata description | |
|---|---|
| Current software version | 2.2.0 |
| Permanent link to executables of this version | https://github.com/ScienceofComputerProgramming/SCICO-D-19-00249 |
| Legal Software License | Apache License 2.0 |
| Computing platform/Operating System | Mac OS X, Linux, Microsoft Windows |
| Installation requirements & dependencies | Latest Oracle JDK 8 or OpenJDK 8 |
| If available, link to user manual - if formally published include a reference to the publication in the reference list | https://github.com/chorevolution/CHOReVOLUTION-IDRE/wiki |
| Support email for questions | amleto.disalle@univaq.it |

**Table 2**
Code metadata (mandatory).

| Code metadata description | |
|---|---|
| Current code version | v2.2.0 |
| Permanent link to code/repository used of this code version | https://github.com/ScienceofComputerProgramming/SCICO-D-19-00249 |
| Legal Code License | Apache License 2.0 |
| Code versioning system used | git |
| Software code languages, tools, and services used | Java 8, Eclipse Oxygen.2, EMF 2.12.0, Eclipse BPMN2 Modeler 1.4.2, Eclipse BPEL Designer 1.0.5, Sirius 5.1.1, Apache CXF 3.1.11, Spring Framework 4.1.9 |
| Compilation requirements, operating environments & dependencies | Java 8, Apache Maven 3 |
| If available Link to developer documentation/manual | https://github.com/chorevolution/CHOReVOLUTION-IDRE/wiki/Developer-Guide |
| Support email for questions | amleto.disalle@univaq.it |

## 1. Introduction

The Future Internet [1] promotes a reused-based software development approach by offering the concrete possibility of composing software services provided by different organizations [2].

Service choreographies are a form of decentralized composition that model the external interaction of the participant services by specifying peer-to-peer message exchanges from a global perspective. When third-party (possibly black-box) services are to be composed, obtaining the distributed coordination logic required to enforce the realizability of the specified choreography is a non-trivial and error-prone task. Automatic support is then needed. The need for choreographies was recognized in the Business Process Modeling Notation 2.0 (BPMN2),[1] which introduced *Choreography Diagrams* to offer choreography modelling constructs. Choreography diagrams specify the message exchanges among the choreography participants from a global point of view.

Within the context of the CHOReVOLUTION H2020 EU project,[2] we developed a platform for the generation and execution of scalable distributed applications specified as choreographies. CHOReVOLUTION has been a 3-years EU project, starting on January 1st, 2015. It represented a huge development effort organized into 7 work packages, and distributed over 8 partners from different countries. The development of the platform required a human investment of 340 person months, and led to write about 150.000 lines of code. Specifically, we realized an Integrated Development and Runtime Environment (IDRE) comprising a wizard-aided development environment, a monitoring console, and a back-end for managing the deployment and execution of choreographed systems on the cloud.[3] A significant aspect of the CHOReVOLUTION IDRE is its ability to ease the realization of choreography-based applications by sparing developers from writing code that goes beyond the realization of the internal business logic of the individual services at stake, taken in isolation. When saying "internal business logic", we mean *extra logic* whose code must be written by the developer for implementing the individual functionalities to be offered by those participants for which no existing service can be reused. When saying "taken in isolation", we mean that developers do not need to care about the distributed coordination logic that is required to enforce the interaction flows specified by the choreography. In other words, the distributed coordination logic, which is needed to realize the global collaboration prescribed by the choreography specification, is automatically synthesized by the IDRE. This capability, together with the simplicity of reusing existing services, was appreciated by the CHOReVOLUTION industrial partners in that the approach permits to develop distributed applications according to their daily development practices.

This work is a thorough extension of the tool paper accepted to the International Conference on Coordination Models and Languages (COORDINATION) [3]. Here, we provide the reader with a complete and practical overview of the IDRE and reports

[1] http://www.omg.org/spec/BPMN/2.0.2/.
[2] http://www.chorevolution.eu.
[3] Indeed, more properly, a choreography is not executed: it is enacted when its participants execute their roles. A participant role models the expected interaction protocol that a service should support in order to play the role of the participant in the choreography.

on its evaluation. Foundational theoretical aspects can be found in previous work [4–6]. In the following, we highlight the main differences with respect to more strictly related previous work:

- In the conference paper, we overview the choreography realization process only. In this paper, we also focus on the architectural style imposed for enabling the automated choreography generation, and on the generated software artefacts (e.g., CDs, Adapters, etc.);
- The illustrative example used in this paper is a different case study. It is more complex and complete in that it demonstrates all the peculiar features of the platform.

The remainder of the paper is organized as follows. Section 2 provides background notions and states the problem solved by the CHOReVOLUTION IDRE. Section 3 shows what the IDRE does and gives an overview of the main components of the IDRE. Section 4 presents the IDRE at work on an industrial use case in the Smart Mobility and Tourism domain, and Section 5 concludes the paper.

## 2. Problem statement

Choreography [7,4,8] is an emergent approach to compose and coordinate distributed services. It describes the interactions among the participant services from a global perspective. Differently from orchestration, choreography does not rely on a central coordinator since each involved service knows precisely when to execute its operations and with whom to interact. Thus, choreography is a collaborative effort focusing on the messages exchange among the participants to reach a common goal [9].

Many approaches have been proposed to deal with foundational problems in choreography design [10–17,7,18]. The need for practical notations for choreography specification was recognized in the OMG's BPMN 2.0 standard, which introduces *Choreography Diagrams*.

In order to put choreographies into practice, we must consider realizing choreographies *by reusing* third-party services. This leads to account for the *automatic realizability enforcement* problem, which can be informally phrased as follows.

Problem statement: *given a choreography specification and a set of existing services, externally* ***coordinate*** *and* ***adapt*** *their interaction so to fulfill the collaboration prescribed by the choreography specification.*

By taking as input a BPMN2 Choreography Diagram, and by exploiting a service inventory where existing services are published so as to be reused for choreography realization purposes, a set of software artefacts are automatically generated in order to implement the adaptation and distributed coordination logic prescribed by the choreography specification. These artefacts adapt and coordinate the interaction among the services – selected as suitable choreography participants – to ensure that their distributed cooperation runs by performing the flows specified in the BPMN2 Choreography Diagram only, hence preventing both interface and interoperability mismatches and the execution of possible flows violating the specification. Furthermore, when needed, specific security policies can be enforced on the participants interaction so to make the choreography secure. These policies concern correct inter-process authentication and authorization. The generated artefacts are:

- **Binding Components (BCs)** serve to ensure middleware-level interoperability among possibly heterogenous services. For instance, a BC can be generated to make a SOAP web service able to communicate with a REST service.
- **Security Filters (SFs)** secure the communication among services by enforcing specified security policies.
- Abstract services defined in the choreography specification characterizes the expected interface of the choreography participants. When realizing a choreography by using the CHOReVOLUTION IDRE, in order to implement the specified choreography participants, concrete services (possibly black-box) are selected from a service inventory and reused. Thus, a concrete service has to match the interface of the participants it has to realize. Here, **Adapters (As)** come into play.
- **Coordination Delegates (CDs)** supervise the interaction among the involved participants to enforce the service coordination logic prescribed by the choreography specification in a fully-distributed way. In other words, CDs act as distributed controllers of the services' interaction.

To better illustrate the problem, Fig. 1 shows a simple example of a BPMN2 Choreography Diagram. Choreography diagrams define the way business participants coordinate their interactions. The focus is on the exchange of messages among the involved participants. A choreography diagram models a distributed process specifying activity flows where each activity represents a message exchange between two participants. Graphically, a choreography task is denoted by a rounded-corner box. The two bands, one at the top and one at the bottom, represent the participants involved in the interaction captured by the task. A white band is used for the participant initiating the task that sends the initiating message to the receiving participant in the dark band that can optionally send back the return message.

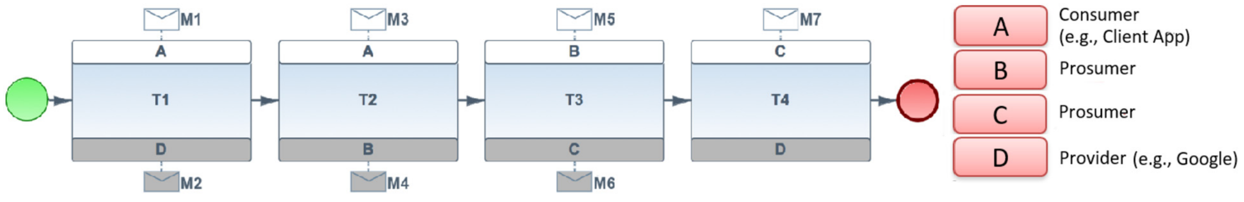


**Fig. 1.** BPMN2 choreography diagram example.

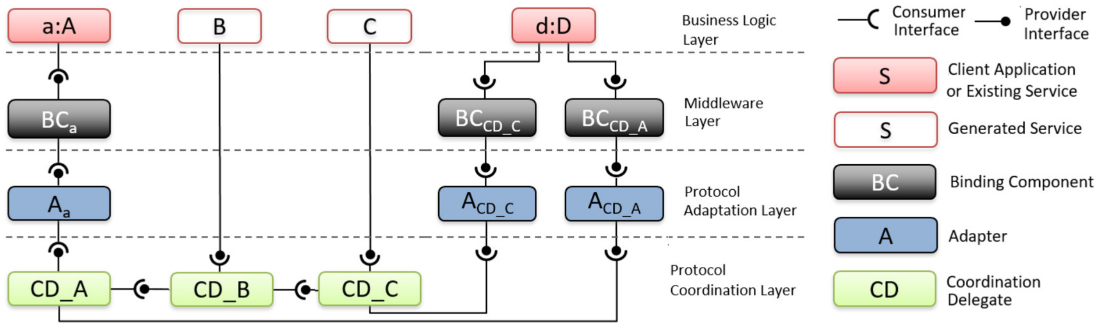


**Fig. 2.** Choreography architectural style (a sample instance of).

The choreography in Fig. 1 involves four participants, A, B, C, and D, for the execution of four sequential tasks, T1, T2, T3 and T4. Specifically, A sends the message M1 to D, enabling it for the execution of T1. After that, D replies to A by sending the message M2. At this point, A sends M3 to B that, after the execution of T2, replies M4 to A and sends M5 to C. Only when M5 is received by C, it executes T3, replies M6 to B and sends M7 to D. Finally, D executes T4 and the choreography ends. By analyzing the choreography, we can distinguish three different types of participants: *consumer*, *provider*, and *prosumer* (i.e., both consumer and provider). For instance, considering a reuse-based development scenario in which existing provider services are published in a suitable service inventory, the consumer participant A might be played by an existing Client App; the provider participant D by an existing Web Service, e.g., Google Maps; B and C might be two prosumers that have to be developed from scratch to realize the specified choreography.

Fig. 2 shows architecture of the system that realizes the choreography specified in Fig. 1, and that is automatically generated by the CHOReVOLUTION IDRE. The top-most layer contains the services representing business logic. In particular, a:A denotes that the role of the consumer participant A is played by a, the Client App in our example; d:D denotes that the role of the provider participant D is played by d, an existing provider service to be reused, whereas, concerning the participant B and C, we do not make use of the notation x:X simply to indicate that they are not existing prosumer services. Thus, they can be either implemented from scratch or partially reused (for the provider part). Then, the second layer contains the BCs to cope with possibly needed middleware-level protocol adaptation, e.g., REST versus SOAP adaptation. It is worth mentioning that SOAP is the default interaction paradigm for the underlying layers. Finally, the last two layers include the Adapter and CD artefacts for adaptation and coordination purposes, respectively. The generated artefacts are not always required; rather, it depends on the specified choreography and the characteristics of the existing services (e.g., application-level interaction protocols, interface specifications, middleware-level interaction paradigms) that have been selected to instantiate the roles of the choreography participants.

## 3. Software framework

This section describes the CHOReVOLUTION IDRE first by giving an overview on what the tool does, and then by introducing its main components along with their implementation.

### 3.1. What the CHOReVOLUTION IDRE does

The CHOReVOLUTION IDRE allows users to (i) define the description models (i.e., interface and security models) of existing services and then publish them in a service inventory; (ii) design a choreography by exploiting BPMN2 Choreography Diagrams; (iii) query a service inventory and select existing services to play the roles of the choreography participants; (iv) define all the details of the interaction among the services involved in the choreography (e.g. service signatures and identity attributes); (v) automatically generate BCs, SFs, As, CDs, Choreography Architecture and Choreography Deployment Description; (vi) automatically deploy and enact the generated artefacts in the cloud according to selectable policies; (vii) monitor the execution of choreography with respect to relevant parameters, such as execution time of choreography tasks, number of messages exchanged for the execution of tasks, end-to-end deadlines, etc., and trigger actions on running services (e.g.,

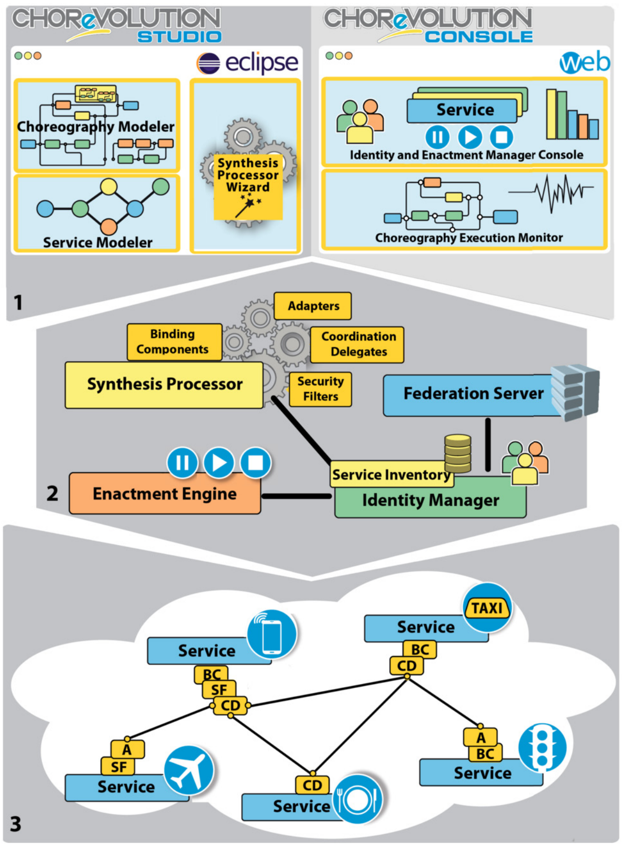


**Fig. 3.** CHOReVOLUTION IDRE overview.

define some authorization policies) and choreographies (e.g., scale up the virtual machines containing the generated artefact related to a choreography).

## *3.2. CHOReVOLUTION IDRE implementation*

As depicted in Fig. 3, the CHOReVOLUTION IDRE is layered into: (1) a front–end layer; (2) a back–end layer; and (3) a cloud layer. The following IDRE components were developed from scratch: the CHOReVOLUTION Studio, the CHOReVOLUTION Console, and the Synthesis Processor together with the artefacts it generates. The other IDRE components were implemented on top of existing open-source projects. For instance, the Identity Manager customizes the Apache Syncope project.[4] It is worth noticing that the choice about the existing projects the IDRE relies on comes from the partners of the CHOReVOLUTION consortium. However, the IDRE is an extensible platform and, as such, in the future, it may also support other technologies such as Kubernetes for deployment and enactment, IBM's AIM for identity management.

**1) The Front–end layer** consists of the following:

The **CHOReVOLUTION Studio** is based on Eclipse Oxygen. The Choreography Modeler is built on top of the Eclipse BPMN2 Modeler plugin and exploits the Eclipse BPEL Designer to allow developers to visualize the generated CDs code. The Eclipse Modeling Framework is used to define the meta-models for Adapters, Choreography Architecture, and the Generic Interface Description Language (GIDL) [19] aptly specified to support Binding Components generation. The Eclipse Sirius project is used to develop the graphical editor of the Choreography Architecture model.

The **CHOReVOLUTION Console** is a web application based on the Apache Syncope and Apache Wicket open source projects.

**2) The Back–end layer** consists of the following:

The **Synthesis Processor** consists of a set of REST (REpresentational State Transfer) services implemented using Apache CXF and the Spring framework. The services employ several model-to-model and model-to-code transformations imple-

[4] https://syncope.apache.org/.

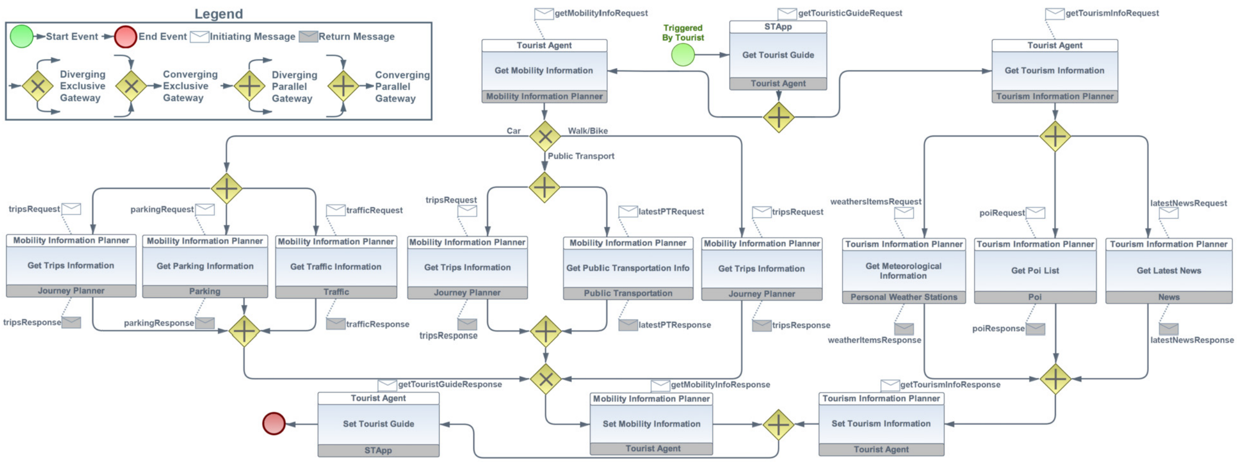


**Fig. 4.** Smart Mobility and Tourism Choreography.

mented in Java by using the following open source frameworks: Apache Freemarker, Eclipse Modeling Framework, Eclipse BPMN2 metamodel, and Ecore BPEL metamodel.

The **Enactment Engine (EE)** is a REST API that extends the Apache Brooklyn project.[5] It automatically deploys the choreography according to its deployment description and listens for command requests from the Identity Manager for runtime choreography control. It is worth noticing that, although choreography monitoring and control is performed by centralized IDRE components (e.g., EE and IdM), the realization and running of the choreography still remain fully distributed into the various artefacts generated by the Synthesis Processor.

The **Federation Server** is a REST service implemented using Apache CXF and the Spring framework. It handles the runtime authentication and authorization for services that uses different security mechanism at the protocol level by storing various credentials on behalf of the caller.

The **Identity Manager (IdM)** is based on the Apache Syncope project, and it is responsible for managing digital identities of users and services. The Service Inventory is a sub-component of the IdM implemented as a REST API based on the extension mechanism provided by Apache Syncope. It acts as a central repository for the description models of the services, such as GIDL (or WSDL), and security policies. The Service Inventory has the same core features as UDDI (Universal Description, Discovery and Integration),[6] (i.e., registry, and search services capabilities), but UDDI only contains WSDL documents. Moreover, UDDI specification is designed to be interrogated by SOAP messages, and it provides more functionalities needed for our purposes.

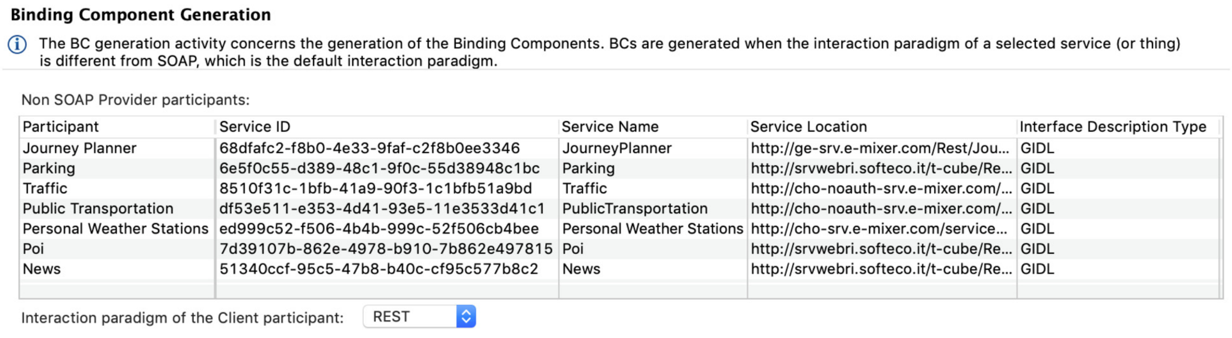
**3) The Cloud layer** executes concrete service choreography instances on a cloud infrastructure and adapts their execution based on the actual application context. At execution time, for each choreography, in the CHOReVOLUTION cloud, there are (i) a set of choreography instances at different execution states; (ii) a set of virtual machines executing a custom-tailored mix of services and middleware components to serve different parts of the choreography. Virtual Machines are installed and configured with services according to selectable policies. Since EE is based on Apache Brooklyn, the CHOReVOLUTION IDRE is not constrained to a specific Infrastructure as a Service (IaaS) platform (e.g., Open Stack,[7] Amazon EC2[8]).

## 4. How to use the CHOReVOLUTION IDRE

This section describes how to use the CHOReVOLUTION IDRE to realize a choreography in the Smart Mobility and Tourism (SMT) domain. Fig. 4 shows the specified BMPN2 choreography diagram. It concerns a Collaborative Travel Agent System (CTAS) in which several content and service providers, organizations and authorities cooperate. The SMT choreography involves a mobile application as an “Electronic Touristic Guide” that exploits CTAS to provide both smart mobility and touristic information.

As shown in Fig. 4, the choreography starts with the mobile application `STApp` detecting the current position of the user, and asking for which type of point of interest to visit and which kind of transport mode to use. From this information, `Tourist Agent` initiates two parallel flows to retrieve the information required by the “Electronic Touristic Guide” (see the parallel branch with two outgoing arrows after the choreography task Get Tourist Guide). In the left-most branch of the

5 https://brooklyn.apache.org/.
6 https://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm.
7 https://www.openstack.org/.
8 https://aws.amazon.com.



**Binding Component Generation**

The BC generation activity concerns the generation of the Binding Components. BCs are generated when the interaction paradigm of a selected service (or thing) is different from SOAP, which is the default interaction paradigm.

Non SOAP Provider participants:

| Participant | Service ID | Service Name | Service Location | Interface Description Type |
|---|---|---|---|---|
| Journey Planner | 68dfafc2-f8b0-4e33-9faf-c2f8b0ee3346 | JourneyPlanner | http://ge-srv.e-mixer.com/Rest/Jou... | GIDL |
| Parking | 6e5f0c55-d389-48c1-9f0c-55d38948c1bc | Parking | http://srvwebri.softeco.it/t-cube/Re... | GIDL |
| Traffic | 8510f31c-1bfb-41a9-90f3-1c1bfb51a9bd | Traffic | http://cho-noauth-srv.e-mixer.com/... | GIDL |
| Public Transportation | df53e511-e353-4d41-93e5-11e3533d41c1 | PublicTransportation | http://cho-noauth-srv.e-mixer.com/... | GIDL |
| Personal Weather Stations | ed999c52-f506-4b4b-999c-52f506cb4bee | Personal Weather Stations | http://cho-srv.e-mixer.com/service... | GIDL |
| Poi | 7d39107b-862e-4978-b910-7b862e497815 | Poi | http://srvwebri.softeco.it/t-cube/Re... | GIDL |
| News | 51340ccf-95c5-47b8-b40c-cf95c577b8c2 | News | http://srvwebri.softeco.it/t-cube/Re... | GIDL |

Interaction paradigm of the Client participant: REST

**Fig. 5.** BC generation activity.

choreography `Mobility Information Planner` is in charge of the retrieval of smart mobility information according to the selected transport mode (see the conditional branching). In contrast, in the right-most branch `Tourism Information Planner` is responsible for gathering touristic information. After that, the two parallel flows are joined together to produce the data needed for the "Electronic Touristic Guide" (see the merging branch with two incoming arrows in the bottom side of the choreography). Finally, the guide is shown to the user employing `STApp`.

The remainder of this section illustrates how to realize the SMT choreography by using the CHOReVOLUTION IDRE. The complete step-by-step user guide to replicate the example can be found at https://github.com/chorevolution/CHOReVOLUTION-IDRE/wiki/User-Guide.

A service provider uses the IDRE to publish the description models of their services into the Inventory. The IDRE allows dealing with the heterogeneity of the services involved in choreography. It provides a uniform description for any service, given by means of the Generic Interface Description Language (GIDL) [19] or the WSDL[9] in case of SOAP services. GIDL supports interface description for any kind of possible services (e.g., REST services). As already said in Section 3, the published services are selected to play the participants roles of a choreography. Then, the next phases will use the services description models to automatically generate the needed BCs, SFs, CDs, and As. For the SMT example, the service provider creates a Service/Thing project and defines the GIDL description for the following services: `Journey Planner`, `Parking`, `Traffic`, `Public Transportation`, `Personal Weather Stations`, `Poi` and `News`. Moreover, the service provider specifies the security model for the service `Personal Weather Stations`.

A choreography developer exploits the CHOReVOLUTION Studio to model a choreography and to realize it. For this purpose, the developer has to create a CHOReVOLUTION Synthesis project. Then, she models the BPMN2 choreography diagram together with the XML messages by using the Eclipse BPMN2 modeler.[10] A message represents the content of a communication between two participants, and BPMN2 designates XML Schema as its default.

After the modelling phase, the developer starts the synthesis process. The first two activities of the process (i.e., Validation and Choreography Projection) do not require any user interaction. The others are supported by suitable wizards as described in the following.

*Selection* - for each provider service of the choreography specification, the developer selects a concrete service from the Inventory. Referring to the SMT choreography, the developer selects all the seven services mentioned above. The current version of the platform does not automate the selection phase. At the end of this section, we discuss how the selection phase will be automated in the next releases.

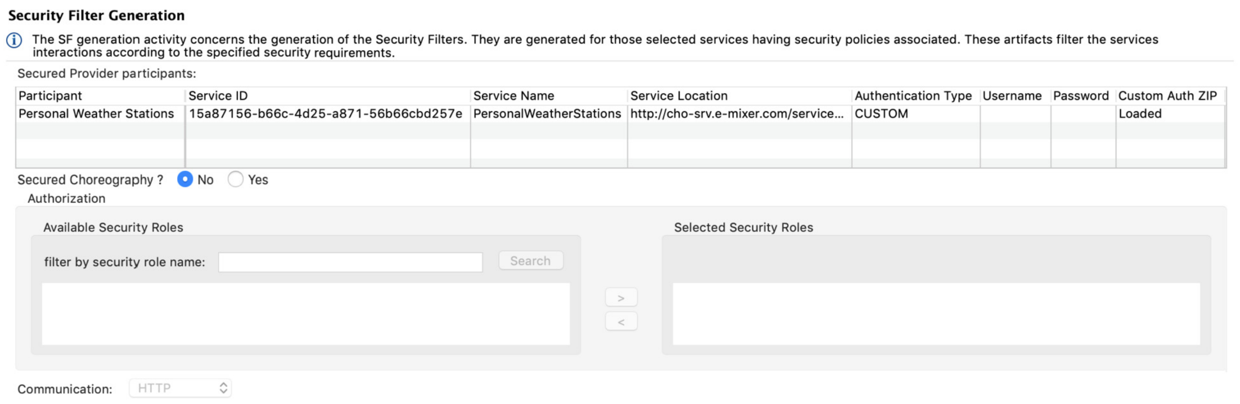
*BC Generation* - Fig. 5 shows the wizard that is used to configure the BCs generator for those selected services that do not rely on SOAP. Moreover, the choreography developer has to specify the interaction paradigm used by the client participant by choosing either REST or SOAP. For the SMT example, since all the services selected in the previous step are REST services, they are listed in the wizard together with their GIDL description. The developer chooses REST as the interaction paradigm of `STApp`.

*SF Generation* - The developer can specify security constraints concerning: (i) the authentication of those selected services that define security policies; (ii) user credentials and authorization rules to access the choreography; (iii) whether the choreography must be accessible using a secured communication protocol (HTTPS) or not (HTTP). For the SMT example, the REST service selected to play the role of `Personal Weather Stations` requires a custom authentication, see Fig. 6.

*Adapter Generation* - Mismatches can arise due to possible heterogeneities between the interfaces of the abstract services and the ones of the selected concrete services. The CHOReVOLUTION Studio is able to solve operation names mismatches and I/O data mapping mismatches. Regarding the SMT choreography, all the selected services exhibit some mismatches with respect to their corresponding choreography participants. The developer can exploit the Adapter Generation wizard to specify the required adaptation logic. The current version of the platform covers operation names mismatches and I/O data mapping

[9] https://www.w3.org/TR/wsdl20-primer/.

[10] https://www.eclipse.org/bpmn2-modeler/.



**Security Filter Generation**

The SF generation activity concerns the generation of the Security Filters. They are generated for those selected services having security policies associated. These artifacts filter the services interactions according to the specified security requirements.

Secured Provider participants:

| Participant | Service ID | Service Name | Service Location | Authentication Type | Username | Password | Custom Auth ZIP |
|---|---|---|---|---|---|---|---|
| Personal Weather Stations | 15a87156-b66c-4d25-a871-56b66cbd257e | PersonalWeatherStations | http://cho-srv.e-mixer.com/service... | CUSTOM | | | Loaded |

Secured Choreography ? [✓ No] Yes

Authorization

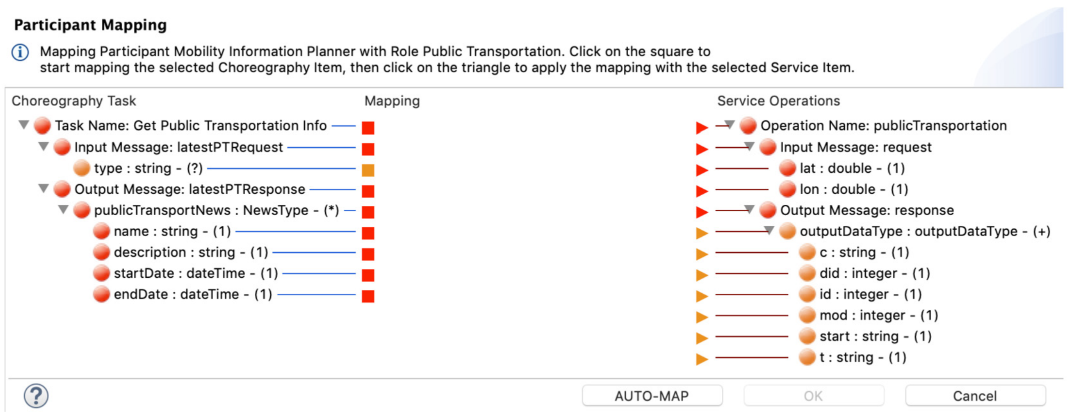
Available Security Roles

filter by security role name:

Selected Security Roles

Communication: HTTP

**Fig. 6.** SF generation activity.

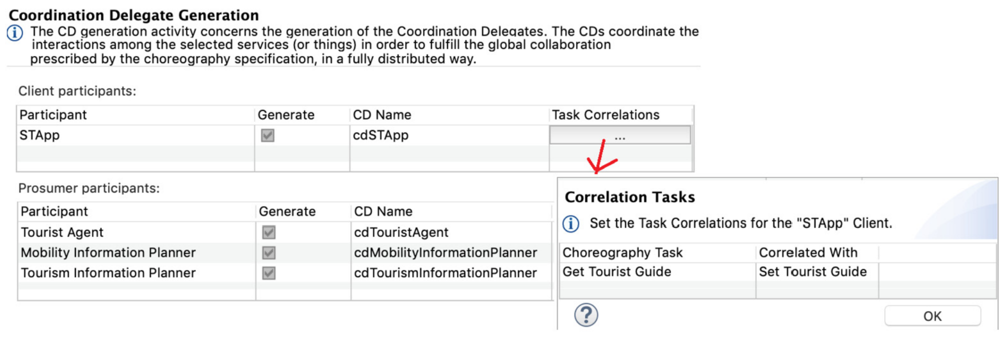


**Participant Mapping**

Mapping Participant Mobility Information Planner with Role Public Transportation. Click on the square to start mapping the selected Choreography Item, then click on the triangle to apply the mapping with the selected Service Item.

| Choreography Task | Mapping | Service Operations |
|---|---|---|
| Task Name: Get Public Transportation Info | 🟥 ▶ | Operation Name: publicTransportation |
| Input Message: latestPTRequest | 🟥 ▶ | Input Message: request |
| type : string - (?) | 🟧 ▶ | lat : double - (1) |
| Output Message: latestPTResponse | 🟥 ▶ | lon : double - (1) |
| publicTransportNews : NewsType - (*) | 🟥 ▶ | Output Message: response |
| name : string - (1) | 🟥 ▶ | outputDataType : outputDataType - (+) |
| description : string - (1) | 🟥 ▶ | c : string - (1) |
| startDate : dateTime - (1) | 🟥 ▶ | did : integer - (1) |
| endDate : dateTime - (1) | 🟥 ▶ | id : integer - (1) |
| | ▶ | mod : integer - (1) |
| | ▶ | start : string - (1) |
| | ▶ | t : string - (1) |

**Fig. 7.** Adapter generation activity. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)



**Coordination Delegate Generation**

The CD generation activity concerns the generation of the Coordination Delegates. The CDs coordinate the interactions among the selected services (or things) in order to fulfill the global collaboration prescribed by the choreography specification, in a fully distributed way.

Client participants:

| Participant | Generate | CD Name | Task Correlations |
|---|---|---|---|
| STApp | ✓ | cdSTApp | ... |

Prosumer participants:

| Participant | Generate | CD Name |
|---|---|---|
| Tourist Agent | ✓ | cdTouristAgent |
| Mobility Information Planner | ✓ | cdMobilityInformationPlanner |
| Tourism Information Planner | ✓ | cdTourismInformationPlanner |

**Correlation Tasks**

Set the Task Correlations for the "STApp" Client.

| Choreography Task | Correlated With |
|---|---|
| Get Tourist Guide | Set Tourist Guide |

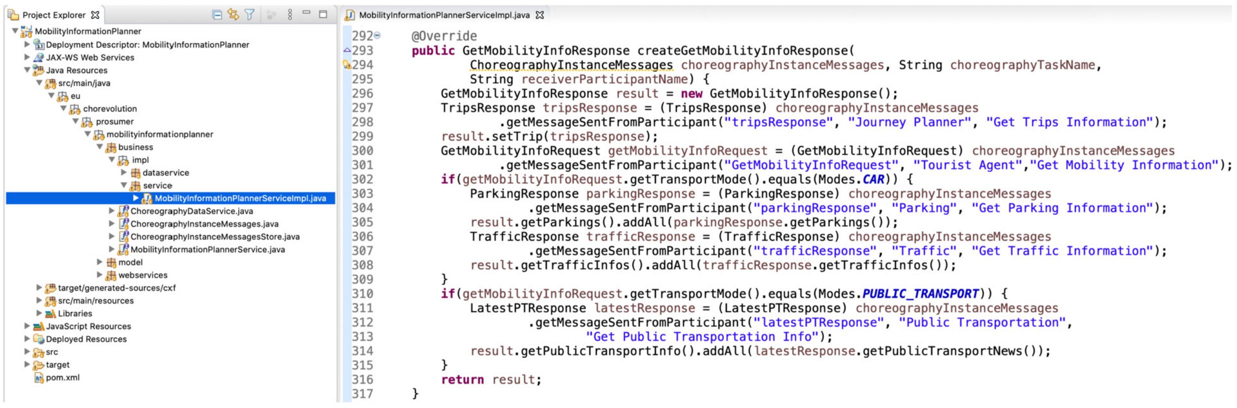
**Fig. 8.** CD generation activity.

mismatches, only. At the end of this section, we discuss how behavioural mismatches can be dealt with. This feature will be offered by next releases together with the automation of the service selection phase.

In particular, the developer maps task messages into service operations messages (Fig. 7). The elements identified with the red shapes are mandatory to be mapped, whereas those in orange are optional. To ease this step, the developer can click the `AUTO-MAP` button to automatically generate a syntactic mapping that can be further refined, e.g., to be semantically correct.

*CD Generation* - The last step of the wizard concerns the automatic CDs generation (Fig. 8).

For the SMT choreography, the mobile application starts the choreography by sending the user preferences (current position, type of transport mode to use, etc.) and finally it gets back all the information needed to show an “Electronic Touristic Guide” to the user. Thus, the developer has to specify a correlation between `Get Tourist Guide` and `Set Tourist Guide`.



```
Project Explorer
▼ MobilityInformationPlanner
  ▶ Deployment Descriptor: MobilityInformationPlanner
  ▶ JAX-WS Web Services
  ▼ Java Resources
    ▼ src/main/java
      ▼ eu
        ▼ chorevolution
          ▼ prosumer
            ▼ mobilityinformationplanner
              ▼ business
                ▼ impl
                  ▶ dataservice
                  ▼ service
                    ▶ MobilityInformationPlannerServiceImpl.java
                ▶ ChoreographyDataService.java
                ▶ ChoreographyInstanceMessages.java
                ▶ ChoreographyInstanceMessagesStore.java
                ▶ MobilityInformationPlannerService.java
              ▶ model
              ▶ webservices
    ▶ target/generated-sources/cxf
    ▶ src/main/resources
    ▶ Libraries
  ▶ JavaScript Resources
  ▶ Deployed Resources
  ▶ src
  ▶ target
    pom.xml
```

MobilityInformationPlannerServiceImpl.java

```java
292     @Override
293     public GetMobilityInfoResponse createGetMobilityInfoResponse(
294             ChoreographyInstanceMessages choreographyInstanceMessages, String choreographyTaskName,
295             String receiverParticipantName) {
296         GetMobilityInfoResponse result = new GetMobilityInfoResponse();
297         TripsResponse tripsResponse = (TripsResponse) choreographyInstanceMessages
298                 .getMessageSentFromParticipant("tripsResponse", "Journey Planner", "Get Trips Information");
299         result.setTrip(tripsResponse);
300         GetMobilityInfoRequest getMobilityInfoRequest = (GetMobilityInfoRequest) choreographyInstanceMessages
301                 .getMessageSentFromParticipant("GetMobilityInfoRequest", "Tourist Agent","Get Mobility Information");
302         if(getMobilityInfoRequest.getTransportMode().equals(Modes.CAR)) {
303             ParkingResponse parkingResponse = (ParkingResponse) choreographyInstanceMessages
304                     .getMessageSentFromParticipant("parkingResponse", "Parking", "Get Parking Information");
305             result.getParkings().addAll(parkingResponse.getParkings());
306             TrafficResponse trafficResponse = (TrafficResponse) choreographyInstanceMessages
307                     .getMessageSentFromParticipant("trafficResponse", "Traffic", "Get Traffic Information");
308             result.getTrafficInfos().addAll(trafficResponse.getTrafficInfos());
309         }
310         if(getMobilityInfoRequest.getTransportMode().equals(Modes.PUBLIC_TRANSPORT)) {
311             LatestPTResponse latestResponse = (LatestPTResponse) choreographyInstanceMessages
312                     .getMessageSentFromParticipant("latestPTResponse", "Public Transportation",
313                             "Get Public Transportation Info");
314             result.getPublicTransportInfo().addAll(latestResponse.getPublicTransportNews());
315         }
316         return result;
317     }
```

**Fig. 9.** Prosumer Business Logic implementation.

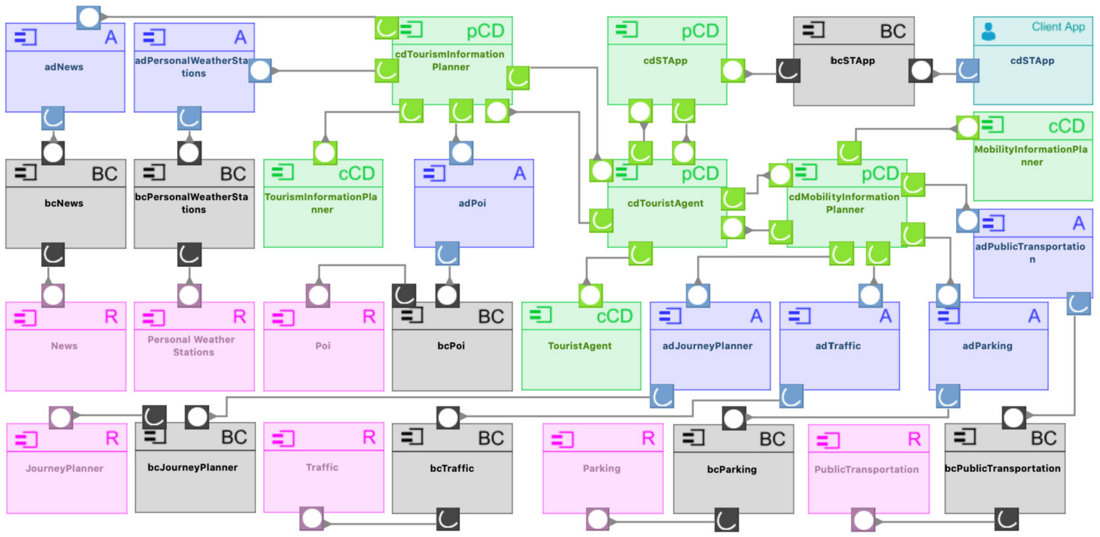
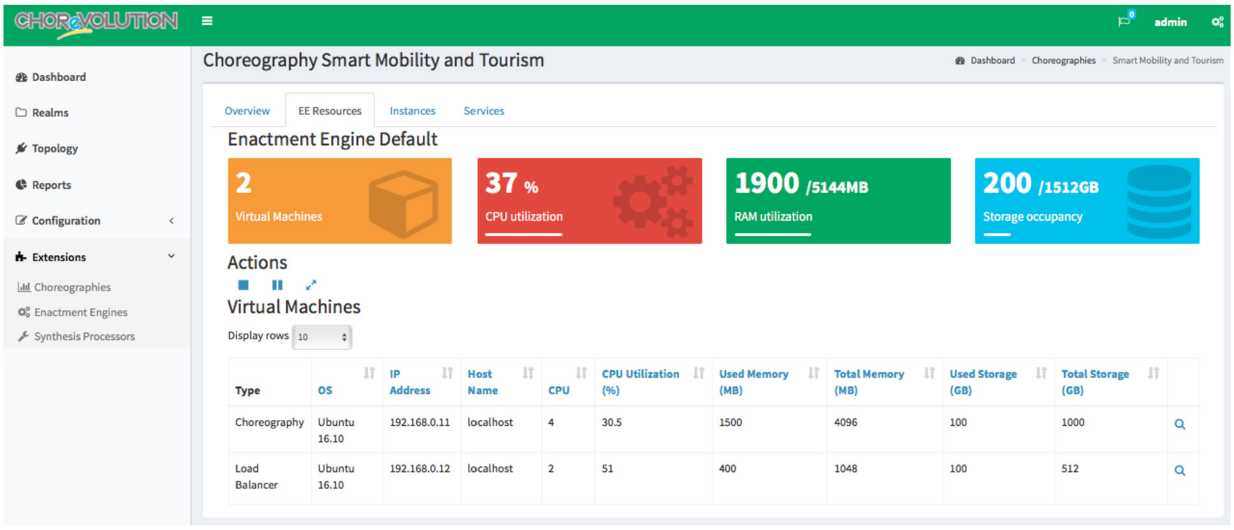


**Fig. 10.** Choreography Architecture.

By clicking on the Finish button, BCs, SFs, ADs, and CDs are generated. Also, for each participant that is an initiating for some task and a receiving for some different task (i.e., Tourist Agent, Mobility Information Planner, and Tourism Information Planner), the skeleton code of its business logic is generated. The developer has to complete it by simply filling in the blanks with the construction logic of the messages sent by the participant. To this purpose, the developer can exploit a repository layer providing the retrieval of previously exchanged messages. The business logic is specified in the `<BusinessLogic>ServiceImpl` class, where `<BusinessLogic>` is the related service name. It is worthwhile noticing that the automated generation/creation of message instances is a semantic and application-specific matter and, as such, its automation cannot be fully achieved in general. Nevertheless, by enhancing the choreography specification with semantics of data, e.g., by exploiting ontologies, ecc., messages generation can be automated to some extent. However, this would require a specification effort, especially in terms of the additional knowledge required to the choreography developer (i.e., the IDRE's user), that goes beyond the daily development activities of an average developer, hence leading one to prefer to complete the generated skeleton rather than learn new specification notations.

Fig. 9 shows the code of the message construction logic for `getMobilityInfoResponse` (see local variable `result`). The implemented logic starts with the retrieval of the message `tripsResponse` sent by Journey Planner within the task `Get Trips Information` (lines 297-298). The content of this message is used to set the trip information of `getMobilityInfoResponse` (line 299). Then, `getMobilityInfoRequest` sent by `Tourist Agent` is retrieved (lines 300-301). Based on the transportation mean chosen by the user, which is contained in the `transportMode` message element, different data can be used to construct `getMobilityInfoResponse`.

*Choreography Architecture Generation* - Considering the selected services and the generated BCs, SFs, ADs, and CDs, an architectural description is automatically generated. Fig. 10 shows the architectural description of the SMT use case. As described in the previous steps, each selected service is a REST service (R purple label) wrapped by a Binding Component (BC black



Choreography Smart Mobility and Tourism

Enactment Engine Default

2 Virtual Machines

37 % CPU utilization

1900 /5144MB RAM utilization

200 /1512GB Storage occupancy

Virtual Machines

| Type | OS | IP Address | Host Name | CPU | CPU Utilization (%) | Used Memory (MB) | Total Memory (MB) | Used Storage (GB) | Total Storage (GB) |
|---|---|---|---|---|---|---|---|---|---|
| Choreography | Ubuntu 16.10 | 192.168.0.11 | localhost | 4 | 30.5 | 1500 | 4096 | 100 | 1000 |
| Load Balancer | Ubuntu 16.10 | 192.168.0.12 | localhost | 2 | 51 | 400 | 1048 | 100 | 512 |

**Fig. 11.** Choreography details page.

label) and by an adapter (A dark blue label). The green boxes correspond to the generated Coordination Delegates. The blue box represents the mobile application.

*Choreography Deployment Description Generation* – The last activity concerns the generation of the Choreography Deployment Description (aka `ChorSpec`) out of the architectural description. The generation is quite straightforward, and after this step, the choreography developer can upload the choreography specification into the Identity Manager. Then, by using the `ChorSpec` and specific deployment policy, the Enactment Engine automatically deploy all the generated BCs, As, SFs, CDs on the cloud (e.g., Amazon EC2). Once correctly enacted, the user can check the health of the virtual machines running the choreography. The choreography details page reports monitoring data collected from each virtual machine, and this data can be used to adapt the virtual machine pool to the expected load (Fig. 11). Further information about the usage of the CHOReVOLUTION console can be found at https://tinyurl.com/quztcay.

**Discussion -** So far, the selection phase is manual because it was not essential to have an automatic selection within the CHOReVOLUTION project. In essence, the Service Inventory, and its related organization and management, was not in the scope of the project and, hence, the current release of the IDRE do not provide automated selection facilities. However, by accounting for service description models that go beyond interface and security policies/constraints such as interaction protocol models (e.g., state machines, automata, Labeled Transition Systems, UML activity diagrams, BPMN2 process diagrams, BPEL, etc.), the IDRE can automatically check whether the interaction protocol (i.e., the activity flow) performed by an abstract choreography participant “match” the interaction protocol of a concrete service in the inventory, which will be then selected to play the role of the participant in case the check succeeds. This protocol equivalence check can be automated by exploiting a suitable notion of (i) participant projection to extract the protocol of a participant out of the choreography specification, and (ii) trace equivalence among interaction protocols, such as the ones we formalized in [4], a work published after the end of the EU CHOReVOLUTION project. In this direction, the Inventory should be extended to (1) allow service providers to publish also the interaction protocol, as further service description model; (2) take as input the interaction protocol of a participant, though choreography projection; and (3) implement the above-mentioned protocol check via trace equivalence, as described in [4]. We are currently work on a new release of the IDRE, where the selection phase is automated according to the previous considerations. Concerning details about the EE, the reader can refer to [17]. For space reasons, in this paper, we preferred to keep the focus on automated choreography synthesis issues. Since the IDRE allows also the generation of the source code for each choreography realization artefact (CDs, BCs, As, SFs), the developer is free to deploy the choreography implementation in any required execution environment, either in the Cloud or not.

**Evaluation -** The CHOReVOLUTION IDRE has been evaluated through two use cases: Smart Mobility and Tourism (SMT) and Urban Traffic Coordination (UTC). The goal of the experiments was to assess the benefit of the proposed approach against a set of Key Performance Indicators (KPIs). These KPIs aim to measure the time saving for realizing and maintaining/evolving the two use cases with the CHOReVOLUTION approach when compared to the development approaches the partners daily use. The experiments revealed that the CHOReVOLUTION approach has great potential in developing choreography-based systems, and the use case got a full benefit from it. A detailed description of the SMT and UTC experiments is available at: https://github.com/chorevolution/CHOReVOLUTION-IDRE/wiki/Experiments.

## 5. Conclusions

CHOReVOLUTION IDRE is an integrated platform for developing, deploying, executing and monitoring choreography-based distributed applications. A use case, in the Smart Mobility and Tourism domain, has been used to show the CHOReVOLUTION IDRE at work. We evaluated the IDRE against two industrial use cases. During the evaluation, the industrial partners experienced a significant time decrease with respect to realizing the use cases by exploiting their daily development approaches. The results of the experiments indicate that CHOReVOLUTION has a great potential in developing choreography-based applications and the two use cases got a full benefit from it.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This work was supported by the (i) Italian Ministry of Economy and Finance, Cipe resolution n. 135/2012 (project INCIPICT - INnovating CIty Planning through Information and Communication Technologies), (ii) SISMA national research project, which has been funded by the MIUR under the PRIN program (Contract 201752ENYB), and (iii) ConnectPA national research project, funded by Regione Abruzzo under the POR FSER Abruzzo 2014-2020 program.

## References

[1] J.M. Hernández-Muñoz, J.B. Vercher, L. Muñoz, J.A. Galache, M. Presser, L.A.H. Gómez, J. Pettersson, Smart cities at the forefront of the future internet, in: The Future Internet - Future Internet Assembly 2011: Achievements and Technological Promises, 2011, pp. 447–462.
[2] European Commission, Digital Agenda for Europe - Future Internet Research and Experimentation (FIRE) initiative, https://ec.europa.eu/digital-single-market/en/future-internet-research-and-experimentation, 2017.
[3] M. Autili, A.D. Salle, F. Gallo, C. Pompilio, M. Tivoli, CHOReVOLUTION: automating the realization of highly-collaborative distributed applications, in: Coordination Models and Languages - 21st IFIP WG 6.1 International Conference, COORDINATION 2019, Co-Located with 14th Int. Conf. on Distributed Computing Techniques, Denmark, June 17-21, 2019, pp. 92–108.
[4] M. Autili, P. Inverardi, M. Tivoli, Choreography realizability enforcement through the automatic synthesis of distributed coordination delegates, Sci. Comput. Program. 160 (2018) 3–29.
[5] M. Autili, A.D. Salle, F. Gallo, C. Pompilio, M. Tivoli, On the model-driven synthesis of evolvable service choreographies, in: 12th European Conference on Software Architecture: Companion Proceedings, ECSA, 2018, 20.
[6] M. Autili, A.D. Salle, F. Gallo, C. Pompilio, M. Tivoli, Aiding the realization of service-oriented distributed systems, in: Proceedings of the 34th Annual ACM Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019, 2019, pp. 1701–1710.
[7] M. Güdemann, P. Poizat, G. Salaün, L. Ye, Verchor: a framework for the design and verification of choreographies, IEEE Trans. Serv. Comput. 9 (2016) 647–660.
[8] L. Chen, C. Englund, Choreographing services for smart cities: smart traffic demonstration, in: 85th IEEE Vehicular Technology Conference, VTC Spring 2017, Sydney, Australia, June 4-7, 2017, 2017, pp. 1–5.
[9] F. Corradini, F. Fornari, A. Polini, B. Re, F. Tiezzi, A formal approach to modeling and verification of business process collaborations, Sci. Comput. Program. 166 (2018) 35–70.
[10] P. Poizat, G. Salaün, Checking the realizability of BPMN 2.0 choreographies, in: Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012, 2012, pp. 1927–1934.
[11] S. Basu, T. Bultan, M. Ouederni, Deciding choreography realizability, in: Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012, pp. 191–202.
[12] M. Autili, D.D. Ruscio, A.D. Salle, P. Inverardi, M. Tivoli, A model-based synthesis process for choreography realizability enforcement, in: Fundamental Approaches to Software Engineering - 16th International Conference, FASE 2013, Rome, Italy, March 16-24, 2013, pp. 37–52.
[13] M. Carbone, F. Montesi, Deadlock-freedom-by-design: multiparty asynchronous global programming, in: Proc. 40th Symposium on Principles of Programming Languages, 2013, pp. 263–274.
[14] I. Lanese, F. Montesi, G. Zavattaro, The evolution of Jolie: from orchestrations to adaptable choreographies, in: Software, Services, and Systems, 2015, pp. 506–521.
[15] S. Basu, T. Bultan, Choreography conformance via synchronizability, in: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011, 2011, pp. 795–804.
[16] S. Basu, T. Bultan, Automated choreography repair, in: Proc. 19th Int. Conf. on Fundamental Approaches to Software Engineering, 2016, pp. 13–30.
[17] L. Leite, C.E. Moreira, D. Cordeiro, M.A. Gerosa, F. Kon, Deploying large-scale service compositions on the cloud with the CHOReOS Enactment Engine, in: Proc. 13th IEEE Int. Symposium on Network Computing and Applications, 2014, pp. 121–128.
[18] D.H. Hu, Q. Yang, CIGAR: concurrent and interleaving goal and activity recognition, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008, 2008, pp. 1363–1368.
[19] G. Bouloukakis, Enabling Emergent Mobile Systems in the IoT: From Middleware-Layer Communication Interoperability to Associated QoS Analysis, Ph.D. thesis, Inria, Paris, France, 2017.