CHOReVOLUTION: Automating the Realization of Highly–Collaborative Distributed Applications

Marco Autili, Amleto Di Salle(✉), Francesco Gallo, Claudio Pompilio, and Massimo Tivoli

University of L'Aquila, Via Vetoio snc, 67100 L'Aquila, Italy
{marco.autili,amleto.disalle,francesco.gallo,
claudio.pompilio,massimo.tivoli}@univaq.it
http://www.disim.univaq.it

Abstract. CHOReVOLUTION is a platform for the tool-assisted development and execution of scalable applications that leverage the distributed collaboration of services specified through service choreographies. It offers an Integrated Development and Runtime Environment (IDRE) comprising a wizard-aided development environment, a system monitoring console, and a back-end for managing the deployment and execution of the system on the cloud. We describe the platform by using a simple example and evaluate it against two industrial use cases in the domain of Smart Mobility & Tourism and Urban Traffic Coordination.

Keywords: Service choreographies · Distributed computing · Automated synthesis

1 Introduction

The Future Internet [20] reflects the changing scale of the Internet and its trend toward the integration and cooperation of different domains supported by an expanding network infrastructure. It relies on large-scale computing environments that will increasingly be connected to a virtually infinite number of services.

This vision is embodied by reuse-based service-oriented systems, in which services play a central role as effective means to achieve interoperability among parties of a business process, and new systems can be built by reusing and composing existing services.

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

Published by Springer Nature Switzerland AG 2019
H. Riis Nielson and E. Tuosto (Eds.): COORDINATION 2019, LNCS 11533, pp. 92–108, 2019.
https://doi.org/10.1007/978-3-030-22397-7_6

choreographies was recognized in the Business Process Modeling Notation 2.0 (BPMN2) [24], which introduced *Choreography Diagrams* to offer choreography modeling constructs. Choreography diagrams specify the message exchanges among the choreography participants from a global point of view.

The CHOReVOLUTION H2020 EU project[1] develops a platform for the generation and execution of scalable distributed applications that leverage the distributed collaboration of services and things by means of service choreographies. In particular, it realizes an Integrated Development and Runtime Environment (IDRE) that comprises a wizard-aided development environment, a system monitoring console, and a back-end for managing the deployment and execution of the system on the cloud.

The CHOReVOLUTION IDRE makes the realization of choreography-based smart applications easier by sparing developers from writing code that goes beyond the realization of the internal business logic. For "internal business logic" we mean the one related to the provisioning of the single system functionalities, as taken in isolation. That is, the distributed coordination logic, which is needed to realize the global collaboration prescribed by the choreography specification, is automatically synthesized by the IDRE. Thus, while coding, developers can avoid to care about coordination aspects. Furthermore, developers can also more easily reuse existing consumers/providers services. These aspects have been appreciated by the industrial partners in that the approach permits to develop distributed applications according to their daily development practices.

The IDRE is an open-source and free software, available under Apache license. It is available as a ready-to-use bundle by the OW2 consortium from https://l.ow2.org/idrevm, and all the documentation can be found at http://www.chorevolution.eu/bin/view/Documentation/WebHome. The source code is also available at https://gitlab.ow2.org/chorevolution.

The remainder of the paper is organized as follows. Section 2 describes the development approach supported by CHOReVOLUTION. Section 3 describes the components constituting the CHOReVOLUTION IDRE. Section 4 highlights the use of the IDRE through a running example. Section 5 briefly evaluates the two CHOReVOLUTION use cases, and Sect. 6 concludes the paper.

2 CHOReVOLUTION Approach

The CHOReVOLUTION synthesis process consists of a set of core *code generation phases* (see Fig. 1) that takes as input a choreography specification together with a set of existing concrete services as possible candidates to play the choreography roles and automatically generates a set of additional software entities. When interposed among the services, these software entities "proxify" the participant services to externally coordinate and adapt their business-level interaction, as well as to bridge the gap of their middleware-level communication paradigms and enforce security constraints.

[1] http://www.chorevolution.eu.

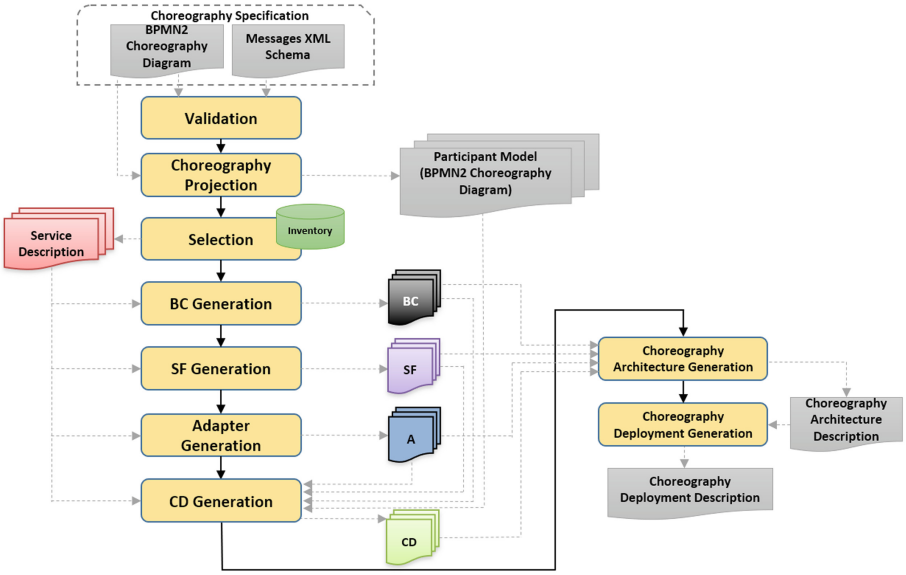

Fig. 1. CHOReVOLUTION development process

Validation – This activity validates the correctness of the choreography specification against the constraints imposed by the BPMN2 standard specification. The goal is to check practical constraints concerning both choreography realizability and its enforceability (see [4,13–15,21–23,26]).

Choreography Projection – Taking as input the BPMN2 Choreography Diagram and the related Messages XML schema, this activity automatically extracts all the choreography participants and applies a model-to-model (M2M) transformation to derive the related Participant Models, one for each participant. A participant model is itself a BPMN2 Choreography Diagram. It contains only the choreography flows that involve the considered participant. The generated participant models will be then taken as input by the Coordination Delegate (CD) Generation activity.

Selection – This activity is about querying the Service Inventory in order to select concrete services that can play the roles of the choreography participants. Once the right services have been selected, the related description models will be used to generate the Binding Components (BCs), Security Filters (SFs), Adapters (As), and Coordination Delegates (CDs).

BC Generation – BCs are generated when the middleware-level interaction paradigm of a selected service is different from SOAP [28], which is used by the CDs as the middleware-level interaction paradigm [16].

SF Generation – SFs are generated for those (selected) services having security policies associated. SFs filter the services interactions according to the specified security requirements.

Adapter Generation – When needed, adapters allow to bridge the gap among the interfaces and interaction protocols of the selected services and the ones of the (respective) participant roles they have to play, as obtained via projection. In other words, adapters solve possible interoperability issues due to operation names mismatches and I/O data mapping mismatches (see [9,10,27]).

CD Generation – CDs are in charge of coordinating the interactions among the selected services so as to fulfill the global collaboration prescribed by the choreography specification, in a fully distributed way (see [5–8,11,12]).

Choreography Architecture Generation – Considering the selected services and the generated BCs, SFs, As, and CDs, an architectural description is automatically generated, and a graphic representation of the choreographed system is provided, where all the system's architectural elements and their interdependencies are represented.

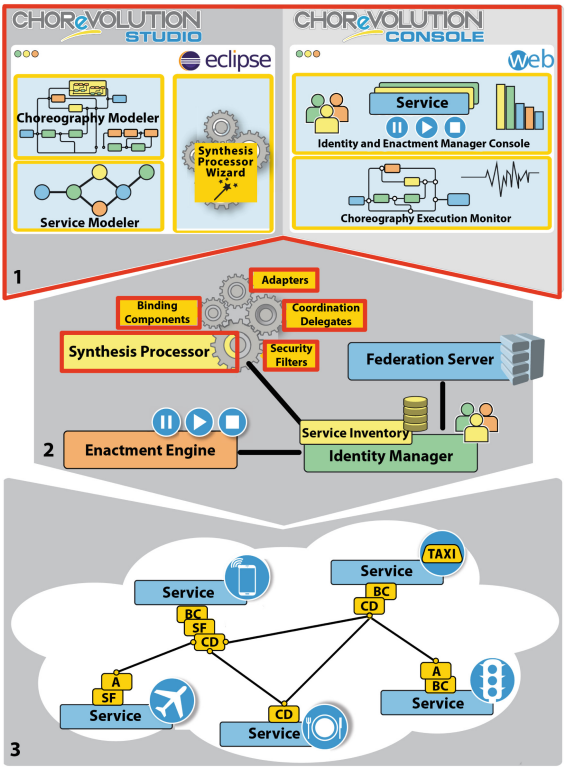

Fig. 2. CHOReVOLUTION IDRE overview

Choreography Deployment Generation – The last activity of the development process concerns the generation of the Choreography Deployment Description (called `ChorSpec`) out of the Choreography Architecture model. The deployment description will be used for deploying and enacting the realized choreography.

3 CHOReVOLUTION IDRE

CHOReVOLUTION IDRE includes software tools for choreography modeling, synthesis, security, identity management and cloud (with monitoring and overall management at run-time).

As depicted in Fig. 2, the CHOReVOLUTION IDRE is layered into: (1) a front–end layer; (2) a back–end layer; and (3) a cloud layer. The red boxes in the figure contain the IDRE components developed within CHOReVOLUTION from scratch. In particular, they are: the CHOReVOLUTION Studio, the CHOReVOLUTION Console, and the Synthesis Processor together with the artifacts it generates. As detailed below, the components outside the boxes are the ones developed within CHOReVOLUTION and built on top of existing open-source projects. For instance, the Identity Manager extends the Apache Syncope project [3]. 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:

(1.1) – The **CHOReVOLUTION Studio** is an Eclipse-based IDE that allows to (i) design a choreography exploiting BPMN2 Choreography Diagrams; (ii) define all the details required to instrument the interaction among the services involved in the choreography (e.g. service signatures, identity attributes and roles); (iii) drive the generation of BCs, SFs, As, and CDs exploiting the automated generation facilities offered by the back–end layer.

(1.2) – The **CHOReVOLUTION Console** is a web-based application that allows to (i) configure, administer and trigger actions on running services and choreographies; (ii) monitor the execution of a 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.

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

(2.1) – The **Synthesis Processor** implements the activities of the synthesis process described in Sect. 2. In particular, it takes as input the BPMN2 choreography diagram and the models of the participant services, and generates all the needed additional software entities that are required to concretely realize the choreography, i.e., CDs, As, SFs, and BCs. Finally, it generates a concrete

description of the choreography (`ChorSpec`) that is passed to the Enactment Engine (via the Identity Manager) for deployment and enactment purposes.

(2.2) – The **Enactment Engine (EE)** is a REST API that extends the Apache Brooklyn project [2]. It automatically deploys the choreography based on the choreography deployment description by using the Cloud Layer. The EE also interacts with the Identity Manager to include into the deployment description the actual deployment and runtime details. Then, once a choreography is deployed and running, the EE 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 artifacts generated by the Synthesis Processor.

(2.3) – The **Federation Server** 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.

(2.4) – The **Identity Manager (IdM)** is based on Apache Syncope project [3] and it is responsible for managing users and services. In particular, the IdM is able to query the services for supported application contexts and played roles; force a specific application context for a certain service (put in "maintenance" or disable/enable). The Service Inventory is a sub-component of the IdM. It acts as a central repository for the description models of the services and things that can be used during the synthesis process.

(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. VMs are installed and configured with services according to selectable policies. Due to the fact that 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 [25], Amazon EC2 [1]).

The IDRE mainly targets three types of users described as follows.

Service providers interact with the CHOReVOLUTION Studio to define the description models (i.e., interface and security models) of existing services and then publish them into the Service Inventory. The benefit they obtain is to foster and ease the reuse of their services by developers, hence increasing the opportunities to be involved in new businesses.

Choreography developers interact with the CHOReVOLUTION Studio to (i) model a choreography by using the Choreography Modeler. (ii) Realize the modeled choreography through the automatic synthesis of BCs (for solving heterogeneity issues), CDs (for solving coordination issues), As (for solving inter-

face mismatches), and SFs (to make the choreography secure). This is done by exploiting the Synthesis Processor.

Choreography operators interact with the CHOReVOLUTION console to (i) deploy and enact the generated choreography-based application through a structured process that involves the back-end layer; (ii) monitor the status of the execution cloud environment; (iii) monitor the execution of the choreography instances and managing their lifecycle.

4 Running Example

This section describes a simple example, called **Chor-eCOM**, that is used as a guide-through to explain the CHOReVOLUTION IDRE. The simple case study falls within the e–commerce domain. It concerns the purchasing of one or more products of different nature.

In our example, the customer is connected to the Internet through a web client or a mobile app. The customer can select a list of items and, once the payment has been performed, the order can be processed, and the invoice can be sent back to her. Then, according to the delivery schedule, the items are packaged and sent to the customer, who will in turn receive shipping information.

Figure 3 shows the Chor-eCOM example choreography specification by means of a BPMN2 Choreography Diagram. A choreography diagram specifies the way the choreography participants exchange information (messages) from a global point of view. The main element of a choreography diagram is the choreography task (e.g., `Order Products` task on left of Fig. 3). Graphically, BPMN2 diagrams uses rounded-corner boxes to denote choreography tasks. Each of them is labeled with the roles of the two participants involved in the task. The white box denotes the initiating participant that decides when the interaction takes place. A task is an atomic activity that represents an interaction by means of one or two (request and optionally response) message exchanges (`orderRequest`) between two participants (`Order Processor` and `Scheduler`).

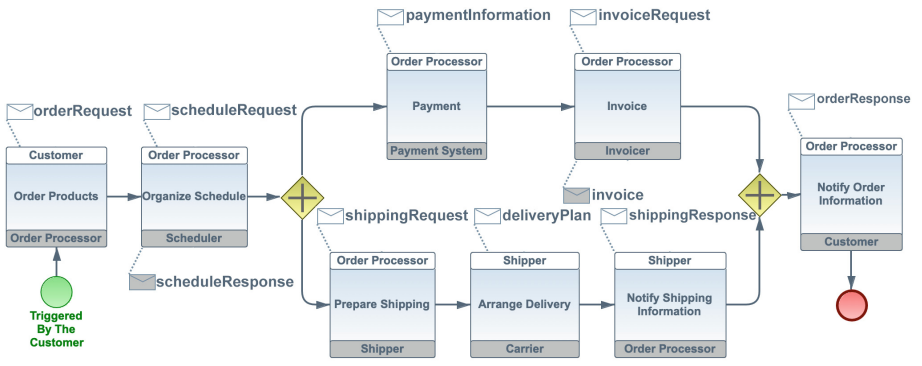

Fig. 3. Chor-eCOM choreography diagram

The use case starts with the mobile application `Customer` sending her information together with the ordering data. From this information, `Order Processor` sends the message `scheduleRequest` to the `Scheduler` participant for scheduling the order. Then, it initiates two parallel flows in order to retrieve the information for delivering and paying the order (see the parallel branch represented as a rhombus marked with a "+", with two outgoing arrows, namely a Diverging Parallel Gateway, just after the choreography task `Organize Schedule`). In particular, the top-most branch retrieves payment and invoice information, while the bottom-most branch gathers delivery information. Finally, the two parallel flows are joined together in order to notify order information to the user by means of `Notify Order Information` choreography task.

We used the choreography specification in Fig. 3 to realize a simple e–commerce choreography–based system where a number of publicly available services can be reused. They have been reused – as black-box third-party software – to instantiate the roles of the participants `Scheduler`, `Payment System`, `Invoicer`, and `Carrier`. The other participants (`Order Processor` and `Shipper`) had to be developed from scratch, and here our synthesis method comes into play. These participants represent the missing logic to be composed and coordinated with the logic offered by the reused services. Note that, the `Customer` participant represents the mobile app used by the user.

In the remainder of this section the Chor-eCOM example is used to show the realization of a choreography-based system by highlighting the roles played by the three types of users of the CHOReVOLUTION IDRE described in Sect. 3.

Service Provider – A service provider uses the IDRE in order to publish the description models of the services into the Service Inventory. The IDRE allows to deal with the heterogeneity of the services involved in a choreography. To this

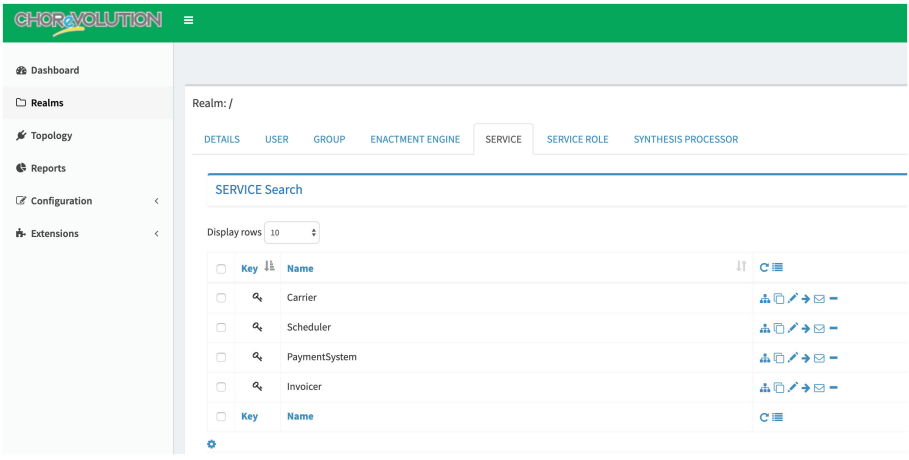

Fig. 4. Service inventory

extent, it provides a uniform description for any service or thing, given by means of the Generic Interface Description Language (GIDL) [16] or the WSDL [29] in case of SOAP services. GIDL supports interface description for any kind of possible services (e.g., REST services) and thing. As already said in Sect. 2, the published services are selected in order to play the participants roles of a choreography. Then, the next phases will use the services models to synthesize the additional software artefacts (i.e., BCs, SFs, CDs, As) of the choreography.

Referring to the Chor-eCOM example, the service provider has to create a Service/Thing project inside the CHOReVOLUTION Studio by using a GIDL description for the following services: `Scheduler`, `Payment System`, `Invoicer`, and `Carrier`. Due to lack of space, we do not provide the steps to create and publish services within the Service Inventory. At the end of this process, the Service Inventory contains the following published services (see Fig. 4).

Choreography Developer – 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, he or she models the BPMN2 choreography diagram together with the XML messages by using the Eclipse BPMN2 choreography modeler [18]. As already discussed in Sect. 2, after the modeling phase, the choreography developer starts the synthesis process. The first two activities of the process (i.e., Validation and Choreography Projection) do not require any user interaction. Then, the choreography developer starts a wizard interface that through several steps realizes the other activities of the synthesis process.

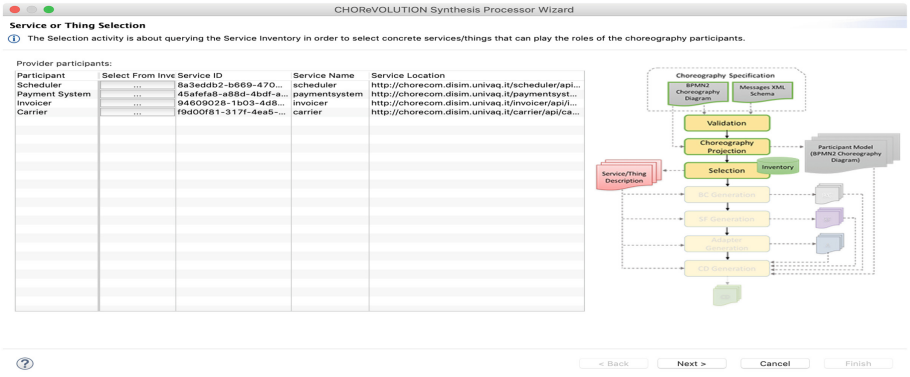

Fig. 5. Selection activity

Selection – The first step of the wizard requires the choreography developer to select, for each choreography participant, its corresponding service from the Service Inventory. Referring to the simple example choreography, the choreography developer has to select all the services published into the Service Inventory as described previously, see Fig. 5.

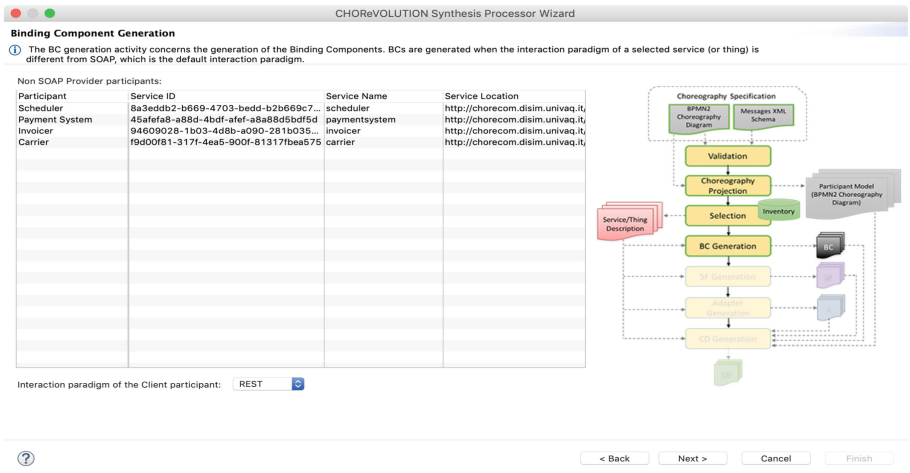

Fig. 6. BC generation activity

BC Generation – Figure 6 shows the step of the wizard used by the choreography developer to configure the Binding Components generator for those selected concrete services that do not use the SOAP [28] protocol. Moreover, the choreography developer has to specify the interaction paradigm used by the client participant of the choreography, by choosing either REST or SOAP.

Considering the Chor-eCOM example, since all the services selected in the previous step are REST services, they are listed in the wizard together with their GIDL description. Moreover, the example provides the purchase information through a mobile application, so the choreography developer has to choose REST as the interaction paradigm of the client participant (`Customer`).

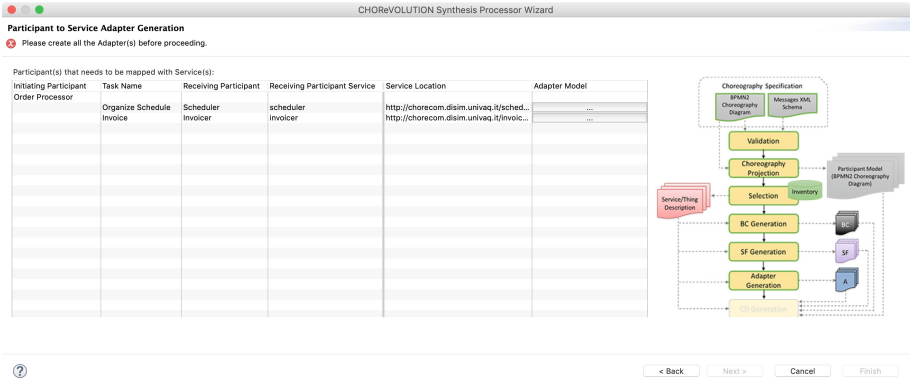

Fig. 7. Adapter generation activity

Fig. 8. Adapter mapping (Color figure online)

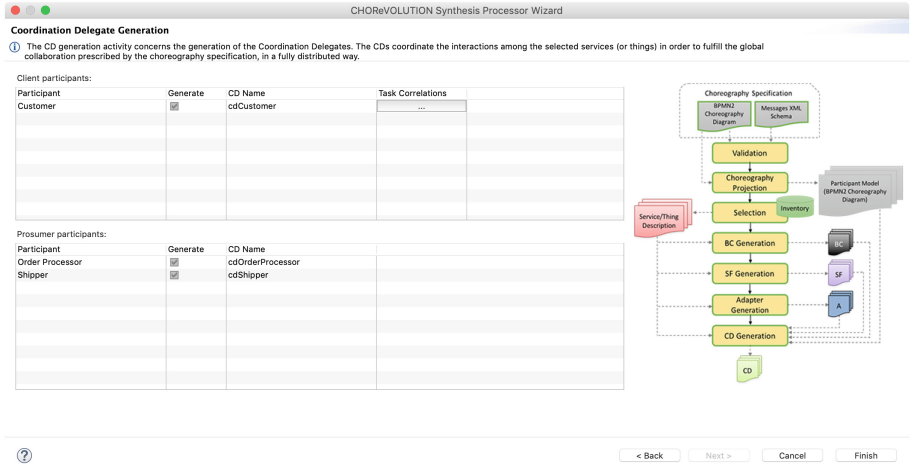

Fig. 9. CD generation activity

The *SF Generation* step is skipped because the existing services do not have security constraints.

Adapter Generation – Figure 7 shows the view of the wizard used by the choreography developer to solve interface mismatches. These mismatches can arise due to a possible gap between the selected services and the (respective) participant roles in the specified choreography. They concern operation names mismatches and I/O data mapping mismatches.

The wizard shows all the choreography tasks that require the choreography developer to specify the adaptation logic, they are grouped by their initiating participant, see the left-most column in Fig. 7. By clicking on the button labeled with "..." a new window is opened, as shown in Fig. 8.

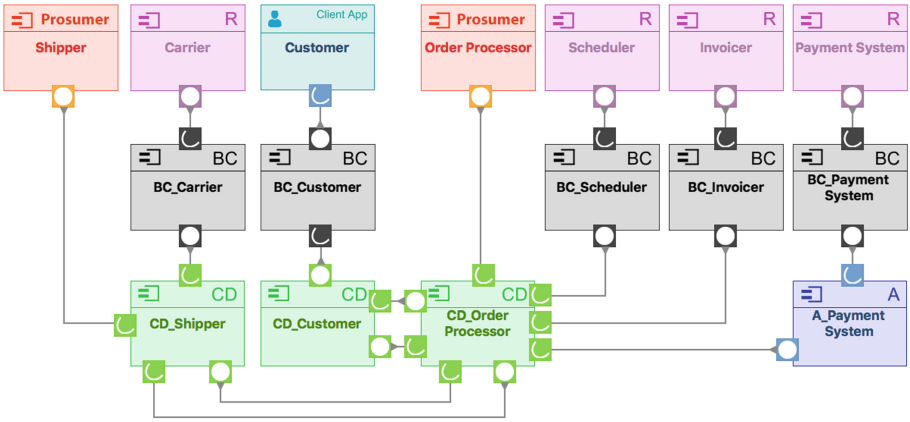

Fig. 10. Choreography architecture (Color figure online)

By interacting with this new dialog window, the choreography developer specifies the mapping between the choreography task messages, reported on the left-most column, and the service operations messages, reported on the right-most column. The elements identified with the red shapes are mandatory to be mapped, whereas those in orange are optional. First, the choreography developer has to map the choreography task with a service operation, and then the related Input and Output messages are auto-mapped. Furthermore, the items forming the message(s) associated to the choreography task under study have to be mapped with the related items forming the specific service message.

CD Generation – It concerns the generation of the Coordination Delegates. The choreography developer has to specify the Correlations Tasks, i.e., a correlation between two choreography tasks. The first task involves a client participant as the initiating participant and a service as the receiving participant. The second task of the correlation involves the same service as the initiating participant and the same client as the receiving participant. In other words, a Correlation Task serves to specify that, in the specified choreography, there are two different tasks that are correlated. Note that these tasks are not necessarily consecutive in the defined choreography flow. This means that the two interactions represented by the two correlated tasks, between the considered client and service, will be indeed realized by a single request-response operation (synchronous interaction) on the service side. The first task corresponds to the invocation by the client of the service' operation and the second task corresponds to the reply by the service to the invocation previously performed by the client.

Considering the Chor-eCOM example, the mobile application starts the choreography by sending the ordering information and then it gets back all the information related to the order. Thus, the choreography developer has to specify a correlation between the task `Order Products` and the task `Notify Order Information` (see Fig. 9).

Choreography Architecture Generation – Considering the selected services and the generated BCs, SFs, ADs, and CDs, an architectural description is automatically generated. A graphical representation (reported in Fig. 10) of the choreographed system shows the architectural description related to the Chor-eCOM example. As described in the previous steps, each selected service is a REST service (R purple label) associated with a Binding Component (BC black label) and an adapter (A dark blue label). The green and red boxes correspond to the generated CDs. The blue box represents the mobile application.

Choreography Deployment Description Generation – The last activity of the synthesis process concerns the generation of the Choreography Deployment Description (aka `ChorSpec`) out of the choreography architectural description. The generation is quite straightforward, and after this step the choreography developer can upload the choreography specification to the Identity Manager.

After the upload of the choreography specification, the choreography is available in the CHOReVOLUTION console (see Fig. 11).

Choreography Operator – At the end of the synthesis phase, the choreography is in the `CREATED` status on the CHOReVOLUTION Console (see Fig. 12). At this point, the Operator can use the "gear" icon to deploy the choreography into the Cloud by passing the `ChorSpec` to the Enactment Engine. After few minutes the status changes to `STARTED`.

Once correctly enacted, the choreography operator can check the health of the virtual machines running the choreography by clicking on the magnifying glass icon. The choreography details page reports monitoring data collected from each virtual machine, and these data can be used by the choreography operator to take action to adapt the virtual machine pool to the expected load, see Fig. 12.

5 CHOReVOLUTION Case Studies Evaluation

CHOReVOLUTION has been evaluated through two use cases: Smart Mobility and Tourism (SMT) and Urban Traffic Coordination (UTC).

The SMT use case has been implemented in cooperation with Softeco, the industrial partner from the Genoa city (Italy). The main scope of the SMT use case is to realize a Collaborative Travel Agent System (CTAS) through the

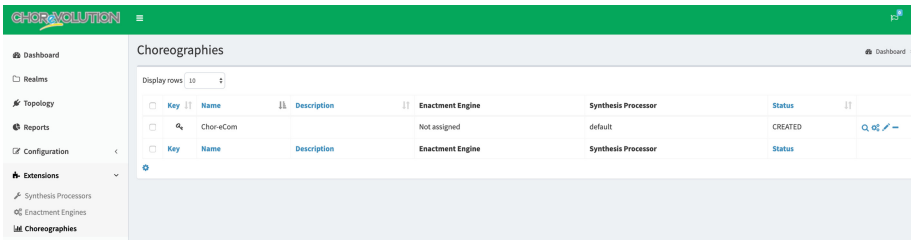

Fig. 11. Uploaded choreography

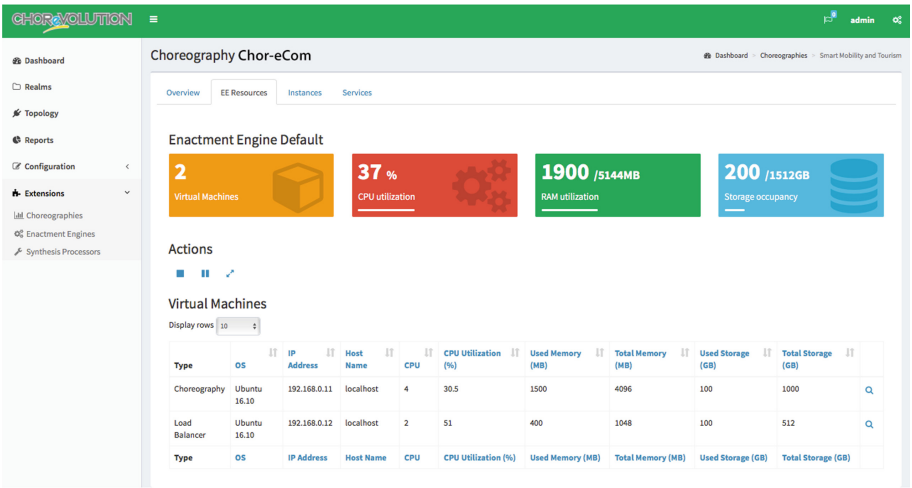

Fig. 12. Monitoring a choreography

cooperation of several content and service providers, organizations and authorities publicly available in Genoa. The SMT use case involves a mobile application as an "Electronic Touristic Guide" that exploits CTAS in order to provide both smart mobility and touristic information.

The Urban Traffic Coordination (UTC) use case has been implemented in cooperation with RISE Viktoria, the industrial partner from the Götheborg city (Sweden). As described in detail in [17], the main scope of the UTC use case is to realize a Cooperative intelligent transport systems (C-ITS) that allows vehicles and transport infrastructure to interconnect, share information and use it to coordinate their actions. The C-ITS provides traffic coordination services exploited through a mobile app for assisting drivers in an eco-friendly and comfortable driving experience.

We evaluated the CHOReVOLUTION IDRE by conducting an experiment for each use case. The goal of the two experiments was to measure the time saving for realizing, maintaining and evolving the two use cases with the CHOReVOLUTION approach when compared to the development approaches the partners daily use. The considered development phases are: **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. According 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.

Table 1. Overall calculation of time savings

EUs	Implementation	Maintenance	Evolution	Time saving
SMT use case				
1	7	0,7	1	–
2	113 **106 saved**	10 **9,3 saved**	27 **26 saved**	**141,3**
3	58,5 **51,5 saved**	8,5 **7,8 saved**	14 **13 saved**	**72,3**
UTC use case				
1	11	0,7	1,5	–
2	152 **141 saved**	12,5 **11,8 saved**	26,5 **25 saved**	**177,8**

The time saving is measured in terms of person-hour (ph). In particular, regarding the SMT use case, we employed the following experimental units:

SMT Experimental unit 1 (SMT-EU1): *CHOReVOLUTION approach* – full usage of the CHOReVOLUTION IDRE except for the development of the mobile application, which is out of the scope.

SMT Experimental unit 2 (SMT-EU2): *General-purpose enterprise-oriented technology* – full usage of the technologies daily adopted by the Softeco partner, i.e., Microsoft .Net, C#, and Visual Studio.

SMT Experimental unit 3 (SMT-EU3): *Domain-specific system integration platform* – full usage of the proprietary platform developed by the Softeco partner, i.e., emixer [19]. It is a content and system integrator that is specific for the travel and mobility information domain.

With respect to the UTC use case, we defined two experimental units. The UTC-EU1 is the same as the SMT-EU1. The second one is defined as follows

UTC Experimental unit 2 (UTC-EU2): *General-purpose enterprise-oriented technology* – full usage of the development technology daily adopted by the RISE partner, i.e., NodeJS and ExpressJS, and Microsoft Visual Studio.

The technologies of SMT-EU2, UTC-EU2, and SMT-EU3 were selected considering that the industrial partners were already skilled with them.

Table 1 summarizes the results of the experiment on the two use cases by distinguishing the implementation, maintenance, and evolution phases. In particular, the EU2 and EU3 highlight in bold the ph saved by using our approach.

The industrial partners experienced a significant time decrease with respect to the their daily development approaches.

6 Conclusions

This paper has presented the CHOReVOLUTION IDRE, an integrated platform for developing, deploying, executing and monitoring choreography-based

distributed applications. A simple explanatory example, 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 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. More pilots and development cases will allow to consolidate the technical maturity of the product and pose the basis for a commercial validation.

Acknowledgments. Supported by: (i) EU H2020 Programme under grant no. 644178 (CHOReVOLUTION - Automated Synthesis of Dynamic and Secured Choreographies for the Future Internet), (ii) the Ministry of Economy and Finance, Cipe resolution n. 135/2012 (INCIPICT), and (iii) the GAUSS national PRIN project (Contract no. 2015KWREMX).

References

1. Amazon: Amazon Elastic Compute Cloud (Amazon EC2). https://aws.amazon.com/ec2/?nc2=h_m1
2. Apache: Apache Brooklyn. https://brooklyn.apache.org/
3. Apache: Apache Syncope. https://syncope.apache.org/
4. Autili, M., Inverardi, P., Tivoli, M.: Automated synthesis of service choreographies. IEEE Softw. **32**(1), 50–57 (2015)
5. Autili, M., Inverardi, P., Perucci, A., Tivoli, M.: Synthesis of distributed and adaptable coordinators to enable choreography evolution. In: de Lemos, R., Garlan, D., Ghezzi, C., Giese, H. (eds.) Software Engineering for Self-Adaptive Systems III. Assurances. LNCS, vol. 9640, pp. 282–306. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-74183-3_10
6. Autili, M., Inverardi, P., Tivoli, M.: Choreography realizability enforcement through the automatic synthesis of distributed coordination delegates. Sci. Comput. Program. **160**, 3–29 (2018)
7. Autili, M., Di Ruscio, D., Di Salle, A., Inverardi, P., Tivoli, M.: A model-based synthesis process for choreography realizability enforcement. In: Cortellessa, V., Varró, D. (eds.) FASE 2013. LNCS, vol. 7793, pp. 37–52. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37057-1_4
8. Autili, M., Ruscio, D.D., Salle, A.D., Perucci, A.: CHOReOSynt: enforcing choreography realizability in the future internet. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, 16–22 November 2014, pp. 723–726 (2014)
9. Autili, M., Salle, A.D., Gallo, F., Pompilio, C., Tivoli, M.: Model-driven adaptation of service choreographies. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, pp. 1441–1450 (2018)
10. Autili, M., Salle, A.D., Gallo, F., Pompilio, C., Tivoli, M.: On the model-driven synthesis of adaptable choreographies. In: Proceedings of MODELS 2018 Workshops: ModComp, Copenhagen, Denmark, 14 October 2018, pp. 12–17 (2018)
11. Autili, M., Salle, A.D., Gallo, F., Pompilio, C., Tivoli, M.: On the model-driven synthesis of evolvable service choreographies. In: 12th European Conference on Software Architecture: Companion Proceedings, ECSA, pp. 20:1–20:6 (2018)

12. Autili, M., Salle, A.D., Gallo, F., Pompilio, C., Tivoli, M.: Aiding the realization of service-oriented distributed systems. In: Proceedings of the 34th Annual ACM Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, 8–12 April 2019, pp. 1701–1710 (2019)
13. Autili, M., Tivoli, M.: Distributed enforcement of service choreographies. In: Proceedings 13th International Workshop on Foundations of Coordination Languages and Self-Adaptive Systems, FOCLASA 2014, pp. 18–35 (2014)
14. Basu, S., Bultan, T.: Choreography conformance via synchronizability. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011 (2011)
15. Basu, S., Bultan, T., Ouederni, M.: Deciding choreography realizability. In: Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012. ACM (2012)
16. Bouloukakis, G.: Enabling emergent mobile systems in the IoT: from middleware-layer communication interoperability to associated QoS analysis. (Systèmes Mobiles Émergents dans l'IoT: de l'Interopérabilité au niveau Middleware de Communication à l'Analyse de la Qualité de Service Associée). Ph.D. thesis, Inria, Paris, France (2017)
17. Chen, L., Englund, C.: Choreographing services for smart cities: smart traffic demonstration. In: 85th IEEE Vehicular Technology Conference, VTC Spring 2017, Sydney, Australia, 4–7 June 2017, pp. 1–5 (2017)
18. Eclipse: Eclipse BPMN2 Modeler, April 2018. https://www.eclipse.org/bpmn2-modeler/
19. EMixer: EMixer. http://www.e-mixer.com
20. European Commission: Digital Agenda for Europe - Future Internet Research and Experimentation (FIRE) initiative (2017). https://ec.europa.eu/digital-single-market/en/future-internet-research-and-experimentation
21. Güdemann, M., Poizat, P., Salaün, G., Ye, L.: VerChor: a framework for the design and verification of choreographies. IEEE Trans. Serv. Comput. **9**(4), 647–660 (2016)
22. Hallé, S., Bultan, T.: Realizability analysis for message-based interactions using shared-state projections. In: Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, pp. 27–36 (2010)
23. Kazhamiakin, R., Pistore, M.: Analysis of realizability conditions for web service choreographies. In: Najm, E., Pradat-Peyre, J.-F., Donzeau-Gouge, V.V. (eds.) FORTE 2006. LNCS, vol. 4229, pp. 61–76. Springer, Heidelberg (2006). https://doi.org/10.1007/11888116_5
24. OMG: Business Process Model And Notation vol 2.0.2, January 2014. http://www.omg.org/spec/BPMN/2.0.2/
25. OpenStack: Open Stack. https://www.openstack.org/
26. Salaün, G., Bultan, T., Roohi, N.: Realizability of choreographies using process algebra encodings. IEEE Trans. Serv. Comput. **5**(3), 290–304 (2012)
27. Di Salle, A., Gallo, F., Perucci, A.: Towards adapting choreography-based service compositions through enterprise integration patterns. In: Bianculli, D., Calinescu, R., Rumpe, B. (eds.) SEFM 2015. LNCS, vol. 9509, pp. 240–252. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-49224-6_20
28. W3C: SOAP Version 1.2, April 2007. http://www.w3.org/TR/soap/
29. W3C: Web Services Description Language (WSDL) Version 2.0, June 2007. https://www.w3.org/TR/wsdl20-primer/