Towards the synthesis of context-aware choreographies

Gianluca Filippone
University of L'Aquila
gianluca.filippone@graduate.univaq.it

Marco Autili
University of L'Aquila
marco.autili@univaq.it

Massimo Tivoli
University of L'Aquila
massimo.tivoli@univaq.it

***Abstract*—Modern technologies and emerging wireless communication solutions in the ICT world are empowering the spread of the most disparate ready-to-use software services distributed over the globe that can be easily accessed by an increasing number of connected devices. This state of affairs offers a dynamic and productive, yet distributed and complex, execution environment that encourages the development of systems based on the reuse of existing services through composition approaches, notably choreographies. However, in order to realize the distributed coordination logic that is required to enforce the correct choreography execution, automatic support is needed. Moreover, environmental changing conditions require the realization of choreographies capable of adapting their behavior to the execution context. This work presents our proposal for addressing the choreography realization problem, by describing an automated process for the synthesis of choreography-based systems capable of performing adaptation according to environmental and context conditions.**

***Index Terms*—architecture synthesis, coordination, adaptation, service choreographies**

I. Introduction

The current trend in the internet and communication fields of ICT is represented by the spread of an increasingly number of technologies and solutions which are leading to a fully connected world. The availability of a virtually infinite number of services, and the advancement of the IoT, are contributing to the connection and the digitalization of our daily lives. Moreover, modern communication technologies like 5G are enabling the coexistence on the network of a very wide range of software services, which can be involved in many different use cases and applications. Smart Cities represent one of the most important outcomes of this process, since they are realized through a complex interaction between services and connected things like intelligent infrastructures, smart vehicles, sensors [1]–[4]. As for Smart Cities, future applications will be built by reusing and composing existing services in a fully distributed fashion. Choreographies represent a powerful and flexible approach for the service composition, in which participants are loosely coupled and the interaction is performed without any central coordinator, allowing services to communicate as peers that autonomously perform tasks according to the global state of the choreography.

Similarly, considering scenarios in which third-party services are reused and composed as black-box entities, the realization of the distributed coordination logic, that is needed to enforce the correct choreography execution, may be a complex and error-prone activity. In fact, the services involved in the choreography act as active entities which concurrently perform tasks and autonomously take decisions. They may not synchronize as prescribed by the choreography, and the global collaboration may not follow the specification. Composing such services and realize a mechanism capable to perform *external coordination* calls for a suitable automated support for choreography developers that can aid their development activities by providing correct-by-construction and ready-to-use solutions to, e.g., concurrency and realizability issues.

Moreover, when dealing with dynamic and large contexts like Smart Cities, systems must be able to deal with the variety of the conditions and technologies exploited to access networked resources. Environmental conditions may change significantly, due to changes of both the network conditions and the services availability. For these reasons, choreography-based systems need to be *context-aware* in order to adapt to changes of the execution context, by varying their behavior and accordingly adjusting the service interaction.

This work aims to extend the approach proposed in our previous work for the automated synthesis of service choreographies [5], [6], in which the collaboration among services is coordinated and enforced by a set of automatically generated software entities, called *Coordination Delegates* (CDs). CDs are interposed among the services involved in the choreography in order to control their interaction. As a novel advance of our previous approach, we propose a solution for the realization of context-aware choreographies. *Variation points* can be used to describe the scenarios that the system can meet according to the possible different context conditions. *Context-aware Coordination Delegates* and synchronization messages allow the choreography to perform runtime adaptation according to the changed situation of the environment.

The paper is structured as follows. Section II briefly recalls background notions coming from our previous approach that are needed to fully understand our novel proposal. Section III sketches our solution for the automated synthesis of context-aware choreographies. Section IV concludes by also discussing future research directions.

II. Background notions

Coordination Delegates (CDs) [5], [6] are additional software entities with respect to the services involved in the choreography. They are automatically generated and properly interposed among the participant services. CDs enforce the

978-1-7281-7735-9/20/$31.00 ©2020 IEEE
DOI 10.1109/ISSREW51248.2020.00072

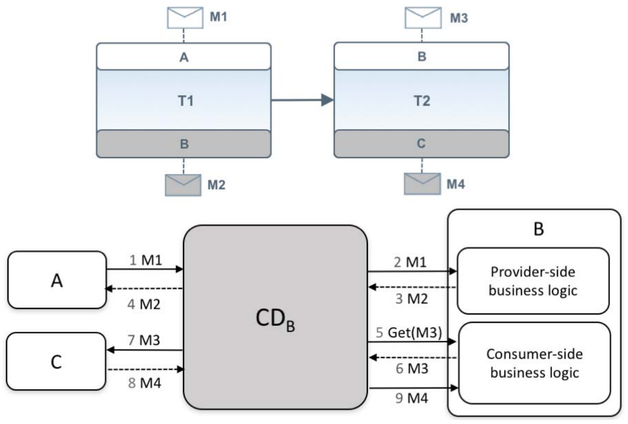

Fig. 1. Coordination delegates interaction pattern

global interaction of the services to behave as prescribed by the choreography specification. They act as service proxies and run a distributed coordination algorithm [6].

CDs are generated for each *prosumer* service[1]. They ensure that the services interaction performs the flows of message exchanges that are described by the choreography specification. CDs allows developers to decouple the coordination logic, that is external to the involved services since it resides in the CDs implementation, from the business logic that instead resides in the services implementation.

Figure 1 shows the CDs interaction pattern by considering the most general sequence of two tasks involving a consumer (A), a provider (C), and a prosumer (B). The prosumer logic is split into *provider-side* and *consumer-side* business logic: the former offers provided functionalities of the service, and can possibly be implemented by reusing an existing service; the latter represents the logic that is needed to build correct message instances whenever the service requires a functionality from a different service, hence performing an invocation of it. As it is shown in the figure, the CD generated for the prosumer B receives the message sent by the service A (M1) as first, hence forwarding it to the provider side of B. Then, the related response messages are sent back, hence ending the execution of task T1 (interactions 1 to 4). In order to execute T2, the CD asks the consumer-side of B for the message to be sent (M3 in the interactions 5 and 6), forwards it to C, intercepts the response and sends it back (interactions 7 to 9).

Following the reuse-based approach in [5], the complete code of CDs, and a skeleton code of their prosumers is automatically generated out of a BPMN2[2] choreography specification.

III. Synthesis of Context-Aware Choreographies

Especially when dealing with dynamic contexts or mission-critical systems, context-awareness capabilities are needed in order to take into account the possibile changing conditions of the execution environment so as to accordingly adapt the choreography. For instance, network performances can change many times while the system is running, or services can become unavailable. Moreover, some operations can involve things and devices that are connected through different and heterogeneous network technologies (e.g., services remotely connected through an optic network, things connected through a mobile 5G network). Network-demanding and business-critical operations should consider the state of both network and services.

In this paper, as notion of *context*, we consider the state and the conditions of the network in terms of performances (e.g., latency, data rate), and the state of the involved services (e.g., availability, current performances). Differently from our previous work, we address context-awareness since the early stages of the choreography realization process, from design and synthesis time to run time.

At design time, we provide novel choreography specification constructs, called "variation points", that allows developer to specify alternative flows of message exchanges for those portions of the choreography that are mission-critical or that can be more affected by context changes. Variation points are accounted for at synthesis time in order to generate an enhanced version of CDs, namely context-aware CDs. Beyond performing external and distributed coordination, at run time, context-aware CDs are able to: (i) sense the current context by dynamically evaluating suitably defined context conditions; (ii) select the right alternative behavior (the right variation) for the choreography; and (iii) finally execute it.

In order to enrich our previous approach with context-aware features, a series of enhancements to the choreography specification, coordination system and synthesis process have to be implemented, as discussed below.

Variation points – Since the above mentioned choreography alternatives do not depend on the content of business messages, nor on the behavior of users and involved services, they cannot be specified by means of usual *alternative flows* of a BPMN2 choreography diagram. Yet, they represent pure *variations* of the choreography and can be properly specified through the definition of *variation points* [7]. However, our concept of variation point differs from the one proposed for SPLs and Systems Families [8]–[10], in the sense that variants do not represent different specifications of a choreography. They are only limited alternative portions of the same choreography, that are dynamically selected and executed at runtime according to the current context conditions.

Variation points are not natively supported by BPMN2. Thus, in order to provide developers with a means to specify them, we extend the BPMN2 metamodel with a novel modeling construct to allow the specification of choreographies like the one shown in Figure 2. It is a sample choreography in which a variation point has been defined. The variation point element is placed – in a form of a new type of task – in a point of the choreography that may be subjected to dynamic adaptation with respect to defined context conditions. It contains the information about the participants involved

[1]A service that is both a consumer and a provider.
[2]https://www.omg.org/spec/BPMN/2.0/

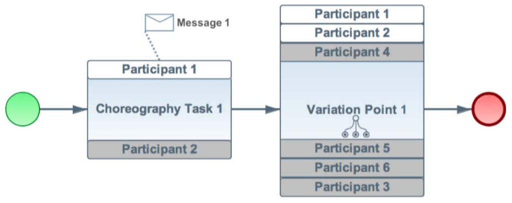

Fig. 2. General case for a variation point

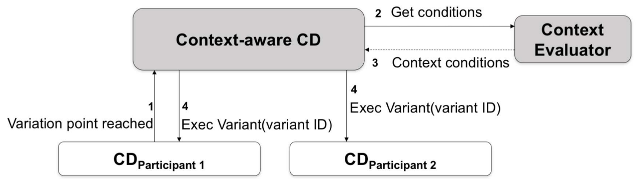

Fig. 3. Context-aware CD interaction pattern

in the alternative variants of the variation point, which are then defined as sub-choreographies. As an extension of the BPMN2 metamodel, variation points can be modeled through a novel ad-hoc metaclass. Variants can be defined as sub-choreographies associated to the new variation point element, while the context can be modeled through appropriated meta-classes that can also allow the specification of the required conditions associated to each variant.

It is worth to note that, while the context conditions required for each variant are defined at design time, the evaluation of the context and the selection of the right variant happen at run time. Moreover, since the interactions among the participant services may change, though relying on well defined alternatives, the whole coordination logic has to take account for the different variants. This requires mechanisms for run-time context evaluation and variation points management.

As a possible application scenario, let us consider an emergency management system of a Smart City. The system composes services to get information about, e.g., building health status, place crowding, traffic. In reacting to an emergency like an earthquake, the system computes an intervention plan according to the structural status of the buildings and the number of people counted into them. In this hypothetical scenario, a variation point can model the different interactions between services for the emergency plan computation, by taking into account the possible unavailability of the place crowding service. A first variant could model the case in which the service is available by including it into the choreography; another variant could consider the unavailability of the service and models the interactions by avoiding the use of the crowding data in the computation of the intervention plan.

Context evaluator – It is associated to a variation point, and is realized as a web service that is in charge of dynamically evaluating the context conditions when the choreography execution reaches the variation point. For instance, the context evaluator can interact with the underlying network layer and get information about its current capabilities and performances, or can check the services' availability. The outcome of this evaluation will be the selection of the right choreography variant to be then executed.

Context-aware CDs – In order to manage the selection and the coordination of a variant, we synthesize a new software entity, called "context-aware CD". As for basic CDs, they can be automatically generated out of the enhanced BPMN2 choreography specification. Context-aware CDs manage variation points by: (i) getting the current context conditions from the context evaluator; (ii) selecting the suitable variant; (iii) informing the other CDs about which variant has to be executed.

Each variation point has a context-aware CD associated to it. Figure 3 shows the interaction pattern that we envision for the context-aware CD, in relation to the variation point in Figure 2. The context-aware CD interacts with the other CDs involved in the task that precedes the variation point (Choreography Task 1). When the variation point is reached, the CD handling Choreography Task 1 notifies the context-aware CD (interaction 1), which asks the context evaluator for the current context conditions (interactions 2 and 3). According to the received response, the context-aware CD selects the variant to be executed and notifies the CDs involved in all the variants (interaction 4). Then, the choreography execution proceeds by performing the selected variant.

We extend the synthesis algorithm presented in [5] by implementing the generation of the context-aware CDs, the context evaluator and an enhancement of the CDs logic in order to handle both the communication with the context-aware CD and the execution of the variants.

Context-aware CDs allow the choreography to adapt to the surrounding execution context. However, in the proposed approach, CDs are aware only of the state of the choreography portion that they coordinate, since execution is enforced by keeping track the sequence of messages exchanged among the participants. CDs only have the strictly needed partial view of the choreography and do not consider the state of other participants nor the state of the global interaction.

Dealing with realizability issues – The cooperation among the participants is driven also by the effects that their actions have on the environment and on the global state of the choreography, rather than by the mere sequence of message exchanges. The choreography specification, in these cases, may not comply with the BPMN2 standard, meaning that it might be not realizable with respect to the realizability notion in [11], [12]. For these reasons, basic CDs would not be adequate to coordinate the services' interaction and to enforce the choreography realizability. Thus, we propose a further refinement of our previous approach by enhancing the dynamic coordination of services.

As analyzed in [6], coordination is needed when *independent sequences* of messages or *independent branches* are contained into the choreography specification. These cases, in BPMN2, are represented by sequences in which the initiating participant of a task does not appear as participant of the im-

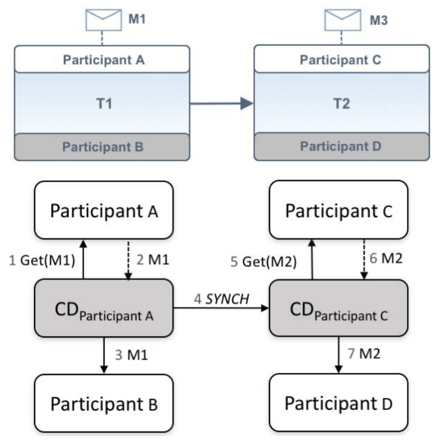

Fig. 4. Coordination of independent sequence of tasks

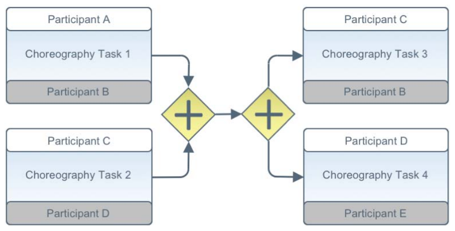

Fig. 5. Independent sequence across a join and a fork gateway

mediately previous task. In order to enforce the choreography realization and deal with independent sequences, it is needed that CDs infer a global view of the system by exchanging coordination messages. They can be exchanged between CDs in a fully distributed way without any central point of control.

Figure 4 shows an *independent sequence* of tasks and its related coordination solution that we propose in this paper. For the sake of generality, let us consider *Participant A* and *Participant C* be two prosumers. We extend the interaction pattern of basic CDs (discussed in Section II) by considering the use of a coordination message (SYNCH) in order to sequentialize the execution of independent tasks, as prescribed by the specification. The SYNCH message is sent from the CD handling the first task ($\text{CD}_{\text{Participant A}}$) to the CD handling the second task ($\text{CD}_{\text{Participant C}}$) as soon as the first task has been accomplished. Note that $\text{CD}_{\text{Participant C}}$ is by construction waiting to receive the SYNCH before performing any other action and, hence, any other possibly concurrent interaction with *Participant C* will be blocked until it can be performed, according to the specification. As formally proved in [6], the overhead due to the exchange of SYCH messages is negligible.

Our novel approach is also able to enforce the coordination of more complex cases, e.g., when the choreography specification includes independent sequences of tasks that go through a fork, a join gateway, or both of them sequentially, as shown in Figure 5. In fact, the same coordination mechanism can be used, by sending SYNCH messages from all the CDs that are handling the tasks that precede the gateway(s) to all the CDs that handle the tasks that immediately follow the gateway(s).

This solution allows CDs to be aware of the state of the choreography also including tasks in which they are not involved, enforcing the choreography to wait for the correct execution state before performing tasks.

IV. Conclusions and Future Works

In this work, we proposed an enhancement of our previous technique for the automated synthesis of service choreographies. The enhancement adds new features for the automated generation of context-aware choreographies. All the new software artifacts can be fully generated from the choreography specification. In particular, we illustrated the notions of variations points (and of choreography variants), context evaluator and context-aware CDs that, all together, allow for the realization of choreographies capable of dynamically adapting to changes in the state and conditions of the underlying network infrastructure and involved services.

Future works concern a formal definition of: (i) the BPMN2 extensions regarding variation points and context conditions; (ii) the SYNCH coordination primitives; and (iii) the algorithm for the synthesis of context-aware CDs and context evaluators. Moreover, in order to validate the context-awareness capabilities of the synthesized choreographies, we plan to show the approach at work on real use-case scenarios, accounting also for highly-dynamic situations in which context changes might happen while executing a previously selected variant, hence calling for the definition of suitable recovery policies.

References

[1] J. Hernández-Muñoz, J. Vercher, L. Muñoz, J. Antonio Galache, M. Presser, L. Gómez, and J. Giæver, “Smart cities at the forefront of the future internet,” pp. 447–462, 05 2011.
[2] L. Purohit and S. Kumar, “Web services in the internet of things and smart cities: A case study on classification techniques,” *IEEE Consumer Electronics Magazine*, vol. 8, pp. 39–43, 2019.
[3] M. R. Palattella, M. Dohler, L. A. Grieco, G. Rizzo, J. T. andThomas Engel, and L. Ladid, “Internet of things in the 5g era: Enablers, architecture, and business models,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, 2016.
[4] L. Chettri and R. Bera, “A comprehensive survey on internet of things (iot) toward 5g wireless systems,” *IEEE Internet of Things Journal*, 2020.
[5] M. Autili, A. Di Salle, F. Gallo, C. Pompilio, and M. Tivoli, “Aiding the realization of service-oriented distributed systems,” in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ACM.
[6] M. Autili, P. Inverardi, and M. Tivoli, “Choreography realizability enforcement through the automatic synthesis of distributed coordination delegates,” *Science of Computer Programming*, vol. 160, 10 2017.
[7] T. Nguyen, A. Colman, M. Adeel Talib, and J. Han, “Managing service variability: State of the art and open issues,” pp. 165–173, 01 2011.
[8] M. A. Babar, L. Chen, and F. Shull, “Managing variability in software product lines,” *IEEE Software*, vol. 27, pp. 89–91, 94, 05 2010.
[9] D. L. Webber and H. Gomaa, “Modeling variability in software product lines with the variation point model,” *Science of Computer Programming*, vol. 53, pp. 305–331, 12 2004.
[10] B. Mohabbati, M. Hatala, D. Gašević, M. Asadi, and M. Bošković, “Development and configuration of service-oriented systems families,” in *Proceedings of the 2011 ACM Symposium on Applied Computing*, p. 1606–1613, Association for Computing Machinery, 2011.
[11] S. Basu, T. Bultan, and M. Ouederni, “Deciding choreography realizability,” vol. 47, pp. 191–202, 01 2012.
[12] S. Basu and T. Bultan, “Automated choreography repair,” in *Fundamental Approaches to Software Engineering*, pp. 13–30, Springer Berlin Heidelberg, 2016.