Towards Adaptable and Evolving Service Choreography in the Future Internet

Amleto Di Salle*, Paola Inverardi* and Alexander Perucci*
* *Department of Information Engineering Computer Science and Mathematics*
University of L'Aquila - ITALY
{*amleto.disalle, paola.inverardi*}*@univaq.it, alexander.perucci@gmail.com*

***Abstract*—The Future Internet is becoming a reality, providing a large-scale computing environments where a virtually infinite number of available services can be discovered and composed so to fit users' needs. In order to enable this large-scale and evolvable computing environment, the ability to automatically compose and dynamically coordinate heterogeneous software services is of paramount importance. Service choreographies are an emergent Service Engineering (SE) approach to compose together and coordinate services in a distributed way. They represent a global specification of the interactions between the participant services. Choreographies will play a central role in Future Internet as an effective means to allow heterogeneous services to suitably collaborate. This *new ideas paper* briefly describes the experience of choreography development that we have been doing so far within the CHOReOS project, and proposes the novel idea we are currently investigating within the IDEAS project to achieve choreography adaptation and evolution through complex data mappings.**

***Keywords*-Service Choreography; Service Oriented Architecture; Distributed Coordination; Protocol Adaptation; Protocol Adaptation; Future Internet;**

I. Introduction

The Future Internet is going to become a reality, providing a large-scale computing environments where a virtually infinite number of available services can be discovered and composed so to fit users' needs. The Internet of Services paradigm emerges from the convergence of the Future Internet (FI) and the service-oriented software paradigm. Services play a central role in this vision as effective means to achieve interoperability between heterogeneous parties of a business process, and new value added service-based systems can be built as a *choreography* of available services discovered within the FI. Service choreography is a decentralized approach, which provides a loose way to design service composition by specifying the participants (i.e., roles) and the (message-based) interaction protocol between them, by decoupling the participant tasks from the services that only later will be bound to the specified roles.

This work is supported by the Ministry of Education, Universities and Research, prot. 2012E47TM2 (project IDEAS - Integrated Design and Evolution of Adaptive Systems), and was supported by the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 257178 (project CHOReOS - Large Scale Choreographies for the Future Internet - www.choreos.eu).

The need for service choreography was recognized in the Business Process Modeling Notation version 2.0[1] (BPMN2), which introduced *Choreography Diagrams* to offer choreography-specific modeling constructs. A choreography diagram models peer-to-peer communication by defining a multi-party protocol that, when put in place by the cooperating parties, will permit to reach the overall choreography objectives in a fully distributed way. In this sense, service choreographies are quite different from service orchestrations in which a single stakeholder centrally plans and decides how an objective should be reached through the cooperation with other services.

As first, we briefly describes the experience of choreography development that we have been doing so far within the CHOReOS EU project[2]. Then, being supported by the IDEAS IT project, this paper proposes the novel idea we are currently investigating to achieve choreography adaptation and evolution to face the challenges posed by the heterogeneity of FI services.

From a technical point of view, this calls for dealing with *mismatching service signature* and *interaction protocol*, as well as *on-the-fly service binding*. In particular, to achieve adaptation, the operations signature and the interaction protocol of the concrete services may need to be adapted to the roles to be played in the input choreography model. This requires to implement a suitable notion of matching between protocols by means of *complex data mappings* over both operation names and I/O messages. Protocol refinement techniques must be developed to bridge the gap between the abstract protocol of the choreography participant roles and the protocol of the concrete services. These techniques, together with the ability of dealing with, e.g., appearing and disappearing services at run-time, permit to achieve evolution. That is, after performing the required data mappings, the employment of on-the-fly service binding techniques and a suitable middleware run-time support allow to handle service addition, replacement, and suppression dynamically.

Towards this challenging goal, the core objective the CHOReOS project has been leveraging relevant SOA standards, while making choreography development a fully dynamic and scalable process. To this end, CHOReOS

[1] http://www.omg.org/spec/BPMN/2.0/
[2] http://www.choreos.eu/

978-1-4799-5069-0/14 $31.00 © 2014 IEEE
DOI 10.1109/SERVICES.2014.65

proposed a development process and implemented an Integrated Development and Run-time Environment (IDRE), supporting the following (macro-)activities: (i) a domain-expert requirements specification framework for adaptable QoS-aware highly-scalable choreographies; (ii) methods and mechanisms for large scale service base management; (iii) choreography synthesis; (iv) service-oriented middleware for the FI to support choreography execution; and (v) analysis of choreography scalability, governance and V&V.

Section II set the context. Then, although we provide an overview of the entire development process model in Section III, in Section IV we focus on the synthesis process and describe our idea for dealing with choreography adaptation and evolution through protocol coordination, protocol adaptation and related complex data mappings. Being supported by the IDEAS project, the implementation of this idea will permit us to enhance the current approach [2], [3] implemented by the CHOReOS synthesis processor[3] [4].

II. Setting the context

When considering choreography-based service-oriented systems, the following two problems are usually considered: (i) *realizability check* - checks whether the choreography can be realized by implementing each participant so that it conforms to the played role; and (ii) *conformance check* - checks whether the set of services satisfies the choreography specification. In the literature many approaches have been proposed to address these problems (e.g., [5]–[7], [9], [10]). However, by moving a step forward with respect to the state of the art, a further problem worth considering when actually realizing service choreographies by reusing (third-party) services concerns *automatic realizability enforcement*. That is, given a choreography specification and a set of existing services (discovered as suitable participants), externally coordinate and adapt their interaction so to fulfill the collaboration prescribed by the choreography specification.

To address this problem, in previous work in [3] supported by the CHOReOS project (and later enhancements in [2]) we propose to (i) possibly adapt the interaction protocol of those services that have been discovered as potential participants of the choreography and (ii) synthesize the global coordination logic to be then distributed and enforced among the considered services. Discovery issues are out of scope of this paper. However it is worth mentioning that, whatever discovery process is applied, it is very infrequent to find a service that exactly matches the discovery query. For this reason, in the literature, many approaches have been devised (e.g., [1], [11]) in order to account for an effective notion of similarity, which is an approximative notion. However, in the context of CHOReOS where the full automation of the realizability enforcement process was one of the main objectives, we could not rely on services that approximately/partially play a choreography role. Thus, our final goal is to synthesize and use adaptors in order to solve the problem of choreography realizability by enforcing exact similarity between the interaction protocol of the discovered services and the one of the choreography roles.

To reach this goal what is missing in [2], [3] is the ability to also perform automatic protocol adaptation by applying complex data mappings over both operation names and related I/O messages.

[3] http://www.choreos.eu/bin/view/Documentation/Development_Environment

[4] http://choreos.disim.univaq.it/

III. CHOReOS Synthesis Process

As shown in Fig. 1, the development process implemented by the CHOReOS IDRE consists of a set of activities, and related artifacts, that allow: (i) to specify requirements by using a dedicated tool; (ii) to derive an initial choreography specification from them, given in the form of BPMN2 choreography diagrams; (iii) to refine the initial choreography specification to allow for automation; (iv) to concretize the choreography specification by discovering services suitable to play the roles of the choreography; (v) to assess the quality of the choreography; and (vi) to automatically synthesize a distributed choreographer, as a set of additional software entities called Coordination Delegates (CDs). The latter are then used to enact and execute the choreography on top of the CHOReOS middleware.

In the remaining of this section, we discuss the importance of complex data mappings (at the level of service operations and their I/O messages) for the synthesis processor while deriving CDs and related adaptors. When put together, CDs and adaptors deal with automatic protocol *coordination* and *adaptation* in a distributed way, respectively.

The synthesis process uses dedicated model transformations to generate out of a BPMN2 choreography an automata-based specification of the coordination logic "implied" by the choreography. Specifically, an extension of Labelled Transition Systems (LTSs), called *Choreography LTS* (CLTS), is generated to explicitly describe the coordination logic that must be applied to enforce the choreography. CLTSs represent the mean to precisely describe the complex coordination logics implied by BPMN2 choreography specifications.

For the choreography to be externally enforced, the coordination logic modeled by the CLTS is distributed between a set of CDs, whose goal is to coordinate (from outside) the interaction of the participant services in a way that the resulting collaboration realizes the specified choreography. To this aim, our method automatically derives the strictly needed CDs, and interpose them among the participant services according to the CHOReOS architectural style (see Fig. 2). At this stage (without considering the adaptors A1, A2, and A3), CDs are able to perform pure coordination of the services' interaction protocol, i.e., *standard communication*

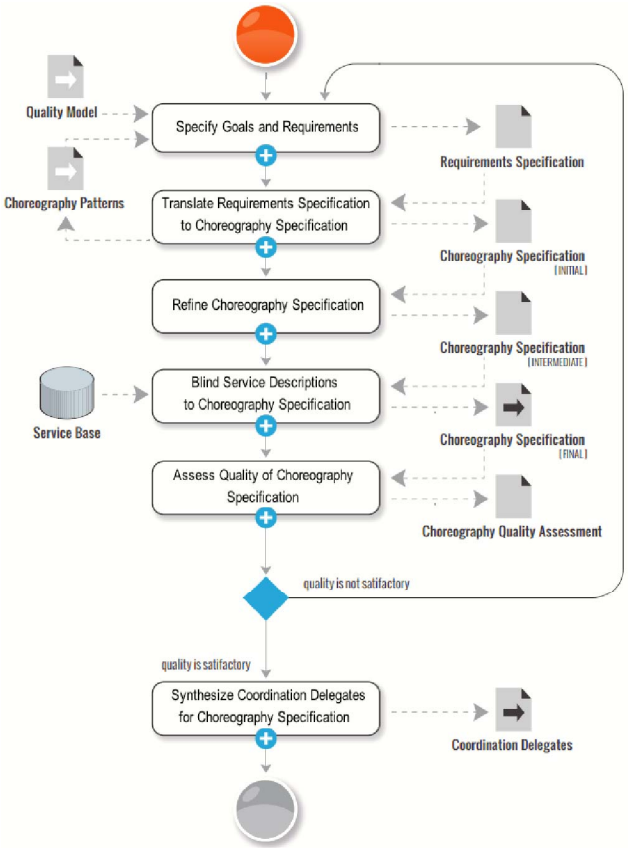

Figure 1. CHOReOS Development Process (first level view)

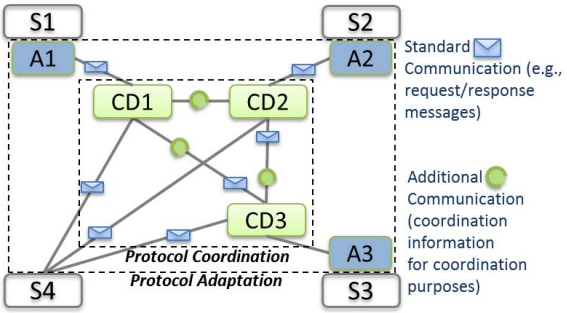

Figure 2. CHOReOS Architectural Style

in the figure. Pure coordination does not account for the I/O messages of the service operations.

The coordination logic is distributed among a set of *Coordination Models* that codify coordination information. Then, at run time, the CDs exchange this coordination information (i.e., *additional communication*) to prevent possible undesired interactions. The latter are those interactions that do not belong to the set of interactions allowed by the choreography specification and can happen when the services collaborate in an uncontrolled way.

In order to understand the notion of undesired interactions let us consider the very simple example in Fig. 3. The figure shows a BPMN2 choreography specification (a) and its corresponding CLTS (b). In BPMN2, a choreography

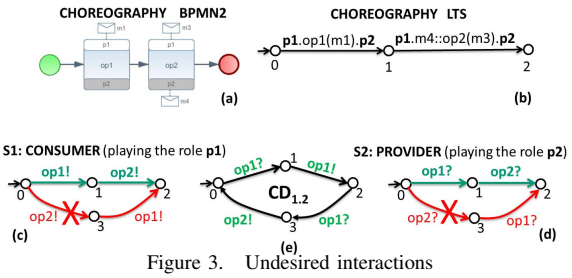

Figure 3. Undesired interactions

`Task` is an atomic activity that represents an interaction by means of one or two (request and optionally response) message exchanges between two participants. Graphically, BPMN2 choreography 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 (see $p1$ and $p2$), and the name of the service operation (see $op1$ and $op2$) performed by the initiating participant and provided by the other one. A role contained in the white box denotes the initiating participant ($p1$ in the two tasks).

The simple BPMN2 model in the figure specifies that the task $op1$ is followed by the task $op2$. The corresponding CLTS models this sequence of tasks by specifying two contiguous transitions. In particular, the transition label $p1.m4{::}op2(m3).p2$ specifies that the participant $p1$ initiates the task $op2$ by sending the message $m3$ to the receiving participant $p2$ which, in turn, returns the message $m4$. Let us now assume that $S1$ and $S2$ are the services that have been discovered to play the roles of $p1$ and $p2$, respectively.

Figure 4. Protocol Coordination

The automaton of $S1$ ($S2$) specifies that $S1$ ($S2$) initiates (receives) $op1!$ ($op1?$) as first, and initiates (receives) $op2!$ ($op2?$) as second, and vice versa. That is, if the services $S1$ and $S2$ interact by following the flow $op1 \rightarrow op2$, the choreography is fulfilled. Vice versa, if the services $S1$ and $S2$ interact by following the flow $op2 \rightarrow op1$, the choreography is not respected. That is, the interaction flow $op2 \rightarrow op1$ is an undesired interaction since, differently from what is specified by the choreography CLTS, the task $op2$ is performed before the task $op1$.

As shown in Fig. 4, the coordination delegate $CD_{1.2}$ is automatically synthesized and interposed between $S1$ and $S2$, proxyfies $S2$ in order to prevent this interaction.

IV. Enhancing Choreography Adaptation and Evolution through Complex Data Mapping

Following the simple example in the previous section, it is easy to see that (*i*) $S1$ and $S2$ can play the roles of $p1$ and $p2$, and that (*ii*) "pure" coordination is enough to achieve correct collaboration, by virtue of the fact that both $S1$ and

$S2$ strongly simulate[5] the roles of $p1$ and $p2$, respectively. This means that the single $CD_{1.2}$ in in Fig. 4 is enough and no adaptor is required.

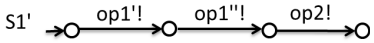

Figure 5. $S1'$: a service discovered to play the role of $p1$

To understand this assertion, let us now suppose that another service, say $S1'$, would have been discovered instead of $S1$. The interaction protocol of $S1'$ is shown in Fig. 5. Let us also suppose that the sequence $op1'! \rightarrow op1''!$ (in $S1'$) is semantically equivalent[6] to (yet syntactically different from) $op1!$ (in $S1$). Thus, the interaction protocol of $S1'$ can be adapted to the one of $p1$ by synthesizing an adaptor (see $A_{1.2}$ in Fig. 6) that merges the sequence of operations $op1'! \rightarrow op1''!$ into the single message $op1!$. This adaptor is synthesized as a wrapper for $S1'$. Note that the same adaptor could be used in the case we would initially consider a service that behaves exactly as $S1$ and is then replaced by $S1'$. This points out how this king of modularity aids evolution in response of possible changes in the participant services, while still keeping the prescribed coordination.

Figure 6. Protocol Adaptation

Another important observation is that, if the I/O messages are not taken care of during the synthesis process, this kind of adaptation is possible only if we assume having a preestablished notion of compatibility, which is always automatically applied, e.g., both $op1'$ and $op1''$ has no return message and the input message of $op1'$ (say $m1'$) merged with the input message of $op1''$ (say $m1''$) can be safely passed to the operation $op1$ offered by $S2$. Basically, with this assumption, the code of the adaptor $A_{1.2}$ might be automatically synthesized so that it accepts the call to $op1'(m1')$, places $m1'$ in a local buffer and immediately return the control to $S1'$, subsequently, accepts the call to $op1''(m1'')$, merges $m1'$ with $m1''$ into a message of type $m1$ (say $m1$), and finally calls $op1(m1)$ on $S2$.

Unfortunately, if we relax this assumption, accounting for only the operation names and their order is not enough anymore; rather, the fully qualified operation signature must be considered and suitable message data mappings might be required. It is evident that this assumption (or similarly strong assumptions) significantly limits the applicability of the overall approach.

That is, to complicate matters further, let us assume the following fully qualified signatures for $op1$, $op1'$ and $op1''$, respectively[7]:

- *boolean md5keyValidity(String key, String encriptedKey)*
- *String getMd5key(String key)*
- *boolean isValidMd5key(String key, String encriptedKey)*

In this case it is not easy to image how an automatic process might handle this kind of situations where the type of I/O parameters must be dealt with in order to infer suitable mappings. The worst comes if we consider the general case where arbitrarily complex data mappings might be required.

Towards pushing ahead with automation of the CHOReOS synthesis process, beyond automating pure coordination, we are investigating within the IDEAS project on how to enhance protocol adaptation through complex data mappings at the (lower) level of service operations and their I/O parameters. This means that the adaptors (such as $A_{1.2}$ in Fig. 6) must be able to treat operation traces' co-relations in conjunction with I/O parameter types' co-relations.

The idea is to infer co-relations of both operations and their I/O parameters by taking into account ontological information. In particular, the approach would be that of extending the domain ontology as used in [8] to semantically enrich the service protocol description with both operation trace co-relations and I/O parameter co-relations. To achieve this, domain-specific semantic relations between concepts, in terms of *subsumption* and *aggregation* [4], must be fine grained (and complex) enough to bridge the gap between different operation traces while keeping sematic integrity of I/O parameters.

V. Conclusions and Future Work

In this paper we have briefly described our experience of choreography development within the CHOReOS project (www.choreos.eu). After providing an overview of the entire CHOReOS development process, we have focused on the choreography synthesis process by highlighting how challenging is achieving full automation of protocol coordination and adaptation when composing heterogeneous services.

Our plan is to keep on studying the idea we are currently investigating within the IDEAS project to achieve choreography adaptation and evolution through complex data mappings. The final goal is to verify how many of the theoretical results we are achieving can be concretely implemented and experimented in real case studies.

References

[1] F. Arbab and F. Santini. Preference and similarity-based behavioral discovery of services. In *WS and Formal Methods*, volume 7843 of *LNCS*. 2013.

[2] M. Autili, A. Di Salle, and M. Tivoli. Synthesis of resilient choreographies. In *SERENE*, LNCS 8166. 2013.

[5]Informally, the behavior of $S1$ and $S2$ "covers" the behavior required by the roles of $p1$ and $p2$, respectively [1], [11].

[6]The existence of a Domain Ontology (DO) is assumed here to semantically enrich the interaction protocol description of the considered services [8].

[7]Indeed, the WSDL specification of the service interface should be considered but, for the lake of space, we use the canonical signature for function declaration in OO programming

[3] M. Autili, D. Ruscio, A. Di Salle, P. Inverardi, and M. Tivoli. A model-based synthesis process for choreography realizability enforcement. In *FASE*, volume 7793 of *LNCS*. 2013.

[4] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, 2003.

[5] S. Basu and T. Bultan. Choreography conformance via synchronizability. In *Proc. of WWW '11*, 2011.

[6] S. Basu, T. Bultan, and M. Ouederni. Deciding choreography realizability. POPL. ACM, 2012.

[7] D. Calvanese, G. D. Giacomo, M. Lenzerini, M. Mecella, and F. Patrizi. Automatic service composition and synthesis: the roman model. *IEEE Data Eng. Bull.*, 31(3):18–22, 2008.

[8] P. Inverardi and M. Tivoli. Automatic synthesis of modular connectors via composition of protocol mediation patterns. In *ICSE*, pages 3–12, 2013.

[9] P. Poizat and G. Salaün. Checking the Realizability of BPMN 2.0 Choreographies. In *Proc. of SAC 2012*, 2012.

[10] G. Salaün. Generation of service wrapper protocols from choreography specifications. In *Proc. of SEFM*, 2008.

[11] E. Toch, A. Gal, I. Reinhartz-Berger, and D. Dori. A semantic approach to approximate service retrieval. *ACM Trans. Internet Technol.*, 8(1), 2007.