A Management Life Cycle for Data-Aware Service Choreographies

Michael Hahn, Dimka Karastoyanova and Frank Leymann

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{hahnml, karastoyanova, leymann}@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{HKL16,
  author    = {Michael Hahn and Dimka Karastoyanova and Frank Leymann},
  title     = {A Management Life Cycle for Data-Aware Service Choreographies},
  booktitle = {Proceedings of the 23\textsuperscript{rd} International
               Conference on Web Services (ICWS 2016)},
  year      = {2016},
  pages     = {364--371},
  doi       = {10.1109/ICWS.2016.54},
  publisher = {IEEE Computer Society}
}
```

A Management Life Cycle for Data-Aware Service Choreographies

Michael Hahn, Dimka Karastoyanova and Frank Leymann
Institute of Architecture of Application Systems (IAAS)
University of Stuttgart, Stuttgart, Germany
Email: {hahnml, karastoyanova, leymann}@iaas.uni-stuttgart.de

***Abstract*—This work is motivated by the increasing importance and business value of data in the fields of business process management, scientific workflows as a field in eScience, and Internet of Things, all of which profiting from the recent advances in data science and Big data. We introduce a management life cycle that renders data as first-class citizen in service choreographies and defines the functions and artifacts necessary for enabling transparent and efficient data exchange among choreography participants. The inherent goal of the life cycle, functions and artifacts is to help decouple the data flow, data exchange and management from the control flow in service compositions and choreographies. This decoupling enables peer-to-peer data exchange in choreographies and provides the means for more sophisticated data management and exchange, as well as data exchange and provisioning optimization.**

***Keywords*-Service Choreographies; Choreography Management Life Cycle; Data Flow Optimization; Transparent Data Exchange**

I. Introduction

Service-oriented architectures (SOA) have seen wide spread adoption during the last decade. The concept of composing small, self-contained units of functionality, so-called services, over the network has found application in many other research areas and application domains like Business Process Management (BPM), Cloud Computing, the Internet of Things (IoT), the emerging paradigm of microservices, eScience and in particular scientific workflows. The specification of service compositions can be realized through various modeling notations following either the service orchestrations or the service choreographies paradigm. Service orchestrations are also known as workflows and are modeled from the viewpoint of one party that acts as a central coordinator [1]. In contrast, service choreographies provide a global perspective of the potentially complex conversations between multiple interacting services without relying on a central coordinator. Each party that takes part in the collaboration, a so-called participant, is able to model its conversations with the other parties by specifying corresponding message exchanges with the participants [2], [3]. Participants in a choreography communicate in a direct, peer-to-peer manner without requiring any central coordinator that controls their interaction.

The business process technology has been successfully applied in the field of eScience. Service choreographies are successfully applied as a means to capture collaborations from a global perspective in both the business domain [2], [4], [5] and in eScience for automating computer-based experiments and simulations [6]–[8]. With the advances in the fields of Big Data and IoT the importance and value of data is increasing significantly [9], [10], but the current state of the art in service choreographies, despite some promissing results showcasing performance benefits [11]–[14], fails to provide an adequate solution that allows data to assume its deserved primary role.

Therefore, in order to account for the crucial importance of data in service choreographies, in this paper, we extend the traditional BPM life cycle with data management capabilities and thus provide an approach that allows treating data exchange in choreographies and orchestrations as a first-class citizen. Towards the same goal, we introduce the Transparent Data Exchange (TraDE) middleware layer that allows for efficient and transparent exchange of data between the participants of service choreographies by decoupling their data flow from the control flow.

The paper is structured as follows: Section II presents the management life cycle of data-aware service choreographies based on the hybrid approach of controlling service compositions through centralized workflow engines while conducting the data flow in a distributed, peer-to-peer manner. Based on the life cycle two approaches, top-down versus bottom-up, for the modeling and execution of data-aware service choreographies are discussed, with focus on the TraDE methods and artifacts for data modeling, exchange and optimization. Finally, the paper concludes with related work in Section III, and a summary of our findings together with an outlook on future work in Section IV.

II. Data-Aware Service Choreography Management Life Cycle

In Figure 1, based on the traditional BPM life cycle [15] and its extensions for choreographies in [2], [16], we present the life cycle of data-aware service choreographies (see also [17]). Throughout the description of each of the life cycle phases and their relationships in the scope of both a top-down and a bottom-up approach, our major focus will be on how the *Transparent Data Exchange* methods, we introduce with this work, support data-awareness as a first class citizen. The *TraDE Methods* bundle a set of data-related methods and transformations to support data-awareness throughout

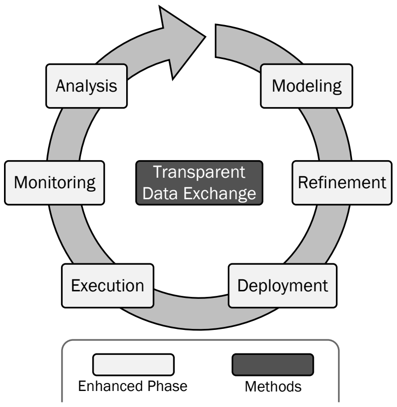

Figure 1. Data-Aware Service Choreography Management Life Cycle [17]

the whole life cycle, as well as potential optimizations on the data perspective of choreographies.

A more detailed view of the extended life cycle and the artifacts each phase uses as input or produces as output, the role of the TraDE Methods, as well as the comparison between the bottom-up and top-down development of choreographies is presented in Figure 2. In our work, we abstract away from any specific terms or constructs through formal models on both the choreography and the process level, and thus maintain the independence of the approach from the various existing modeling languages for choreography and process model specification. This allows us to apply our approach to any choreography and process modeling notation by defining a mapping from the formal to the concrete models. We use the concept of *Choreography Model Graphs* (CM-Graphs) introduced in [18] and *Process Model Graphs* (PM-Graphs) defined in [19] as underlying formal notation (metamodel). PM-Graphs provide a formal foundation and specify process models as directed acyclic graphs, where the nodes represent activities and the edges the control connectors (control flow) between the activities. An activity expresses a task or piece of work to perform, like invoking a service or manipulating data. CM-Graphs are based on the PM-Graph definition and introduce the ability to couple two or more existing PM-Graphs while obfuscating all activities that are not involved in the interconnection (coupling). We use the term *communication activity* to refer to the remaining activities that are performing the conversations between the participants, i. e., activities that are sending or receiving messages. The coupling between the PM-Graphs is specified through a set of directed edges where each edge connects a communication activity from one PM-Graph with a communication activity of another PM-Graph. These special interconnection edges are called *message links* and represent a conversation between two choreography participants. Since the message links also carry the data to be exchanged between choreography participants, the data flow and hence the data exchange is tightly coupled to the control flow between choreographies. Other constructs of the CM-Graph or PM-Graph metamodel, required in the context of this work, are introduced when necessary in the following sections.

Top-down Approach

We assume that during a life cycle loop, in each phase all successor phases have access to the knowledge and artifacts of the current and all predecessor phases, as well as the history of previous life cycle loops. Following the top-down approach means that a new choreography model is specified first, based on which a collection of implementing process models is generated and refined.

A. Modeling

In the *Modeling* phase the stakeholders, e. g., domain experts of different fields in eScience or business specialists from several companies in the business domain, define their interactions by specifying corresponding participants and their conversations (message exchanges) through a choreography model. The choreography model can therefore be seen as a collaboration contract on which all participants agree. *BPEL4Chor* models [20], *Business Process Model and Notation* (BPMN) collaboration models [21], *Web Services Choreography Description Language* (WS-CDL) models[1] or *Let's Dance* models [22] can be used, for example, as underlying modeling notation to represent choreography models. Up till now, the data exchanged between the participants, specified through the collaboration contract, is tightly coupled to the specification of the conversations. In this work, we introduce an explicit data model and data flow between the participants in a choreography. The resulting *Choreography Data Model* (CDM) provides the foundations for the data-awareness of choreographies and supports (semi-)automated phase transitions and model enhancements throughout the life cycle.

The result of the Modeling phase, depicted in part A of Figure 2, is a manually defined choreography model that comprises two or more participants, their interaction logic and a *Choreography Data Model* (CDM). The CDM can be automatically generated based on the modeled message exchanges or manually defined by the modeler. The interaction logic is specified through corresponding communication activities and message links between the participants. The CDM consists of a set of *Data Objects* that are explicitly connected with communication activities, participants or associated to message links through so-called *data links*. A Data Object has a unique identifier and contains one or more *Data Elements*. A Data Element has a unique name from the scope of its surrounding Data Object and a reference to a definition of its data structure, e. g., using XML Schema Definitions[2]. The Data Object can be seen as

[1]W3C, Web Services Choreography Description Language Version 1.0. Online available: http://www.w3.org/TR/ws-cdl-10/

[2]W3C, XML Schema Definition Language (XSD) 1.1 Part 1: Structures. Online available: http://www.w3.org/TR/xmlschema11-1/

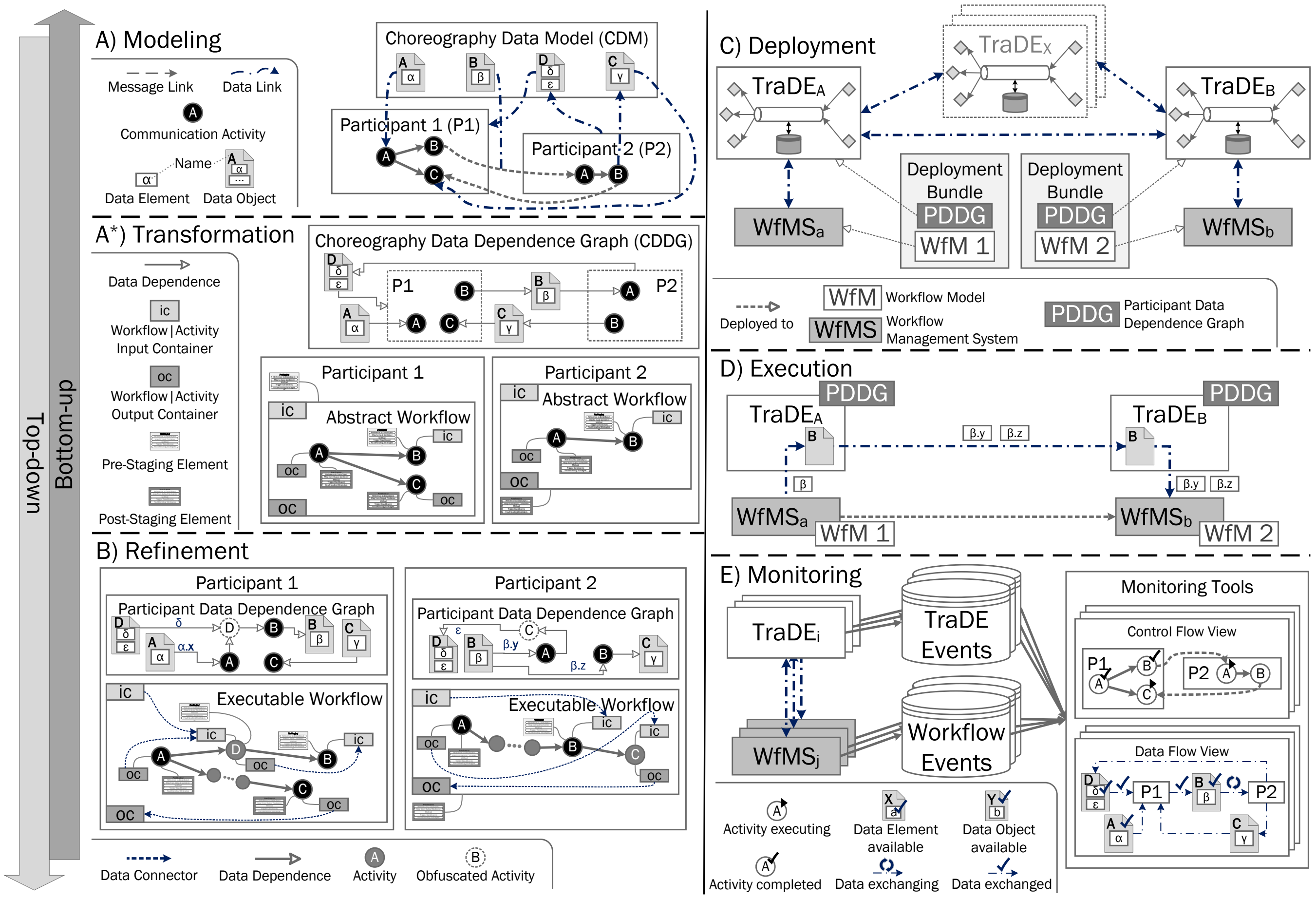

Figure 2. Top-down vs. bottom-up approach for the modeling, enactment and optimization of data-aware service choreographies

a named envelope for a collection of typed containers (Data Elements) that hold the corresponding data values during run time. We distinguish between Data Objects that can be instantiated only once (single-instance) and Data Objects that hold a collection of values (multi-instance) during run time. Multi-instance Data Objects are useful to hold a collection of identically structured values that can be processed with the same sequence of activities through a loop, while the loop counter can be dynamically set to the number of elements of the collection during run time.

At the end of the modeling phase, a *Transformation* step takes place in which a high-level process model (also known as *abstract workflow model*) is generated for each of the specified choreography participants. The generated process models together implement the globally agreed collaboration behavior and are used as templates in the *Refinement* phase. These processes are not directly executable since they lack required details like the internal process logic and the required run time environment configuration data for successful deployment. *BPMN* process models [21] or abstract *Business Process Execution Language* (BPEL) process models [23] can be used to represent these high-level processes. An example of how such a transformation may look like is described in [24] where BPEL4Chor is used as choreography notation and BPEL to express the generated process models.

The transformation itself is performed in three steps. In the *first step* the participants communication logic are transformed from the choreography model into *abstract workflow models*. From the perspective of our underlying metamodel this means that the PM-Graphs contained in the choreography model are split and stored as new independent process models. In the *second step* the specified data dependencies between Data Objects and participants in the choreography model are analyzed using the TraDE methods and the results are collected in a *Choreography Data Dependence Graph* (CDDG). We rely on dependency analysis concepts well known from the domain of programming languages and compiler theory where the control and data dependencies between the statements of a program are identified and expressed in form of graphs to optimize the scheduling and execution of a program [25], [26]. The CDDG is a representation of the data dependencies between the choreography participants and we will use it to enrich the generated process models and to optimize the data placement during deployment and the data staging and exchange during later life cycle phases. A CDDG is a directed graph where the nodes are choreography participants, activities and Data Objects, and the edges the read and write dependencies between them (see part A^* of Figure 2).

During the *last step*, the data-specific transformations will take place, i. e., the data dependencies collected in the CDDG are incorporated into the abstract workflows or PM-Graphs, respectively, in form of so-called *Staging Elements*, resulting in the artifacts shown in part A^* of Figure 2. We distinguish between Pre- and Post-Staging Elements, where the former are triggered before the activity implementation is executed and the latter after the activity implementation is finished. With Pre-Staging Elements data from a Data Object can be written to the input container of an activity or the data from the input container can be written to a Data Object. The Post-Staging Elements define similarly the staging of data between a Data Object and the output container of an activity. For each data dependence edge in the CDDG one Staging Element is generated in a workflow and is associated to the activity or participant which is referenced by that edge.

The structure of Pre- and Post-Staging Elements is identical and contains a *Reference* to a Data Object, a *Map* between the Data Elements of the referenced Data Object and the data elements of the activities' input or output container, a *Staging Method* (e. g., push, pull or transfer) that should be invoked during run time, one or more *Trigger Conditions* that specify under which conditions a Staging Element should be activated and a set of *Fault Handling Strategies*. The staging elements are used to enact the transparent data exchange in choreographies.

B. Refinement

During the *Refinement* phase IT specialists refine the generated high-level process models to make them executable. The result is a collection of executable process models (also known as workflow models) depicted in part B of Figure 2. IT specialists of the different participants introduce new activities and specify the control flow as well as the data flow inside the workflows by connecting activities with so-called *data connectors*. Data connectors are comprised by *data maps* that define the explicit mapping between the data elements of two containers. The data connectors and the corresponding data maps provide important input for our TraDE methods. After the manual refinement is completed we are analyzing the workflow models to extract the new information about the internal data dependencies and the data flow. For each executable workflow model a so-called *Participant Data Dependence Graph* (PDDG) is generated. For this, the CDDG created during transformation is split into subgraphs where each subgraph represents the data dependencies of one participant (PDDG). All activities added during refinement that read or write data from or to a Data Object are added to the PDDG. The resulting initial PDDGs are then refined and extended with the knowledge gained from the analysis of the refined workflows. For example, activity D shown in part B of Figure 2 is such an added activity that reads data from the input container of the workflow which depends on Data Object D. Furthermore, for each data connector between the containers of two activities a corresponding data dependence edge is added to the PDDG, if at least one of the activities depends on a Data Object. The specified data maps are used to detail the dependencies in the PDDG on the level of the Data Elements, i. e., which parts of the Data Object are read or written by an activity. For example, in Figure 2 activity B depends on the data produced by activity D represented by the data connector from the output container of activity D and the input container of activity B. In addition, the data map of activity D shows that it only reads Data Element δ of Data Object D which is also incorporated in the resulting PDDG model as a label on the data dependence edge between the two nodes. After the refined data flow on the level of the workflows is analyzed and the PDDGs are updated, also the model element to which a Staging Element is associated after transformation can be potentially optimized. For example, the Pre-Staging Element associated to Participant 1 which represents the dependence edge from Participant 1 to Data Object D, can now moved to activity D, as depicted in part B of Figure 2, since no other activity of the participant depends on the data. In addition to the described refinements, the IT specialists also specify the required configuration data for the targeted run time environment to enable the deployment of the workflows.

C. Deployment

In the *Deployment* phase the executable workflow models are packaged in the required format and deployed to the target workflow middleware. Depending on the requirements of the collaborating parties the deployment distribution can range from deploying all models to one central workflow middleware, to deploying each model to a different middleware node. We distinguish between static and dynamic strategies for the deployment of the workflows. Therefore, the TraDE methods have to take into consideration all available information, e. g., data dependence graphs, monitoring data or manually defined deployment requirements, to find an optimal deployment distribution. In the context of this work, we are focusing only on the static strategies, whereas dynamic scenarios will be investigated in future.

At the end of the Deployment phase the choreography is deployed and prepared for execution (see part C of Figure 2), i. e., all workflow models and PDDGs are deployed to corresponding WfMS and their associated TraDE nodes, respectively.

D. Execution

The deployed executable workflow models enter the *Execution* phase upon instantiation. With this the choreography that these workflows realize is also being executed. In the following, we use the term *choreography instance* introduced in [27] to describe these groups of interrelated workflow instances, without implying that there is a central entity coordinating the workflow instances according to the choreography model. During the execution of the choreography instance each of the participating workflow instances produces a set of events that provide information about executed activities, control and data flow, occurred exceptions or faults. These events are analyzed through the TraDE methods to detect potential data flow optimizations during the choreography instance execution, e. g., in terms of optimal data placement, transferring data in advance based on predictions calculated from monitoring information or optimal data life cycle management, so that the data is only stored as long as required and as short as possible. Therefore, the TraDE middleware uses the information collected in the PDDGs and the CDDG as well as the Staging Elements together with the event data of previously executed choreography instances. Part D of Figure 2 shows an example in which the PDDG is used to find out which Data Elements of Data Object B have to be exchanged between the TraDE nodes A and B, so that they can be read by the workflow model executed by WfMS$_b$.

E. Monitoring

In order to *Monitor* choreography instances the event data of the involved workflow instances needs to be collected, analyzed, combined and interpreted. For example, the status of the choreography instance has to be calculated through the combination of the status of all workflow instances. The resulting data can be expressed again in form of higher-level choreography events, so that the interpretation and combination has to be done only once and other interested parties are able to directly consume the choreography events instead of interpreting the lower-level workflow events on their own. An environment that enables the monitoring of choreographies is introduced in [28]. For data-aware choreographies the explicitly modeled data flow and the data flow adaptations during run time triggered by optimization have to be monitored, too.

For this reason, since the TraDE middleware helps decouple the data and the control flow of choreographies, data-related events that are emitted by the TraDE middleware have to be defined to allow the monitoring of the data staging, placement and exchange and thus to ensure that the optimized data flow is still carried out according to the choreography models. Correlation of data-related events is an open issue that has to be addressed on the level of WfMS and TraDE middleware. The data-related events are collected per TraDE node in an event database. Since the event data of a choreography is distributed over multiple databases as shown in part E of Figure 2, an overall identifier is required to correlate the events with each other. For example, a choreography instance identifier can be introduced on the level of the WfMS and TraDE middleware which can be used to identify and correlate all events of one choreography instance [28].

F. Analysis

The goal of the *Analysis* phase in conjunction with the modeling phase is to produce choreography and process models that are optimal with respect to a set of requirements. Already existing models from earlier life cycle iterations together with their monitoring data are therefore also taken into account. Established techniques and methods to identify optimization possibilities are, for example, the use of modeling best practices, the detection of so-called anti-patterns [29] or the simulation of model alternatives based on quantitative information (also known as *instrumentation*) [19]. The applied optimizations might have an impact on both the level of the choreography and the process models since the conversations specified in the choreographies are implemented by the processes.

Bottom-up Approach

In contrast to the top-down approach, the goal of the bottom-up approach is to produce a choreography model that reflects the interactions of an already existing and executable collection of interconnected workflow models. The main reasons for using the approach are on the one hand to create a global view of the collaboration behavior and the conversations between the interconnected workflows that can be analyzed and inspected and on the other hand to identify global optimization potentials.

The sequence of abstraction steps to be performed starts with grouping the workflow models into participants in the *Refinement* phase (see part B of Figure 2). For each of the participants a PDDG is generated by analyzing the executable workflows: all their communication activites are identified and added to the PDDG and for each of the messages referenced by the identified communication activities a new Data Object is generated, added to the PDDG and connected with the activity through a data dependence edge. The structure of the Data Objects and their Data Elements are derived from the structure of the related messages. After the generation of PDDGs is completed, IT specialists are allowed to manually refine them, e. g., by adding (obfuscated representations of) additional activites of the underlying workflow to the PDDG and connecting them with the existing Data Objects. Afterwards, each PDDG is used to generate and annotate a collection of Staging Elements to the corresponding workflow model. Again IT specialists are able to manually refine and extend the automatically generated Staging Elements, e. g., by specifying fault handling strategies. Since the workflow models were already executed, we can incorporate the available data and information from the *Monitoring*, *Execution* and *Deployment* phases into the abstraction and refine the generated PDDGs and the Staging Elements or at least provide valuable supporting information to the IT specialist during the manual refinement. Furthermore, the event data of the Monitoring phase can be used, for example, to annotate the control flow edges of the workflows with corresponding probabilities to support predictive data staging in future executions, as described in Section II, or to identify data-related and control-related trigger conditions to enrich the generated Staging Elements.

Following the bottom-up approach, the *Transformation step* produces a set of abstract workflow models in which the internal logic of the participants is abstracted away and only the communication activities contributing to the conversations between participants in the choreography are kept. As shown in part A* of Figure 2, the Staging Elements are simply copied and all parts of the executable workflow models that are not required or should not be visible on the level of the choreography are removed or at least obfuscated. Furthermore, the PDDGs generated during the Refinement phase are merged into one overarching CDDG that represents the data dependencies of all choreography participants. The last automatic abstraction step brings us to the choreography *Modeling phase* where the abstract workflow models are combined into an overall choreography model with corresponding message links between the communication activities of the participating workflows. Therefore, the CDDG is used as input to generate the corresponding Choreography Data Model and connect the Data Objects with the activities as defined by the data dependence edges of the CDDG. After the choreography model is generated, changes can be applied and propagated to the lower layers following the top-down approach again. The changes can therefore be incorporated into the abstract and executable workflow models by selectively updating and merging them. The resulting choreography model can also be used as a template and extended with other participants in different application scenarios.

III. Related Work

In this section, we will compare our approach with some of the existing approaches that focus on decoupling the data flow in service compositions and choreographies from the control flow or ordered message exchange. None of the approaches focuses explicitly on the management life cycle of choreographies.

The model-driven approach presented in [30] supports the modeling and enactment of data exchange in choreographies using messages by extending the BPMN modeling language and introducing annotations for BPMN data objects. The annotations are then automatically transformed into SQL queries to specify and enact message extraction from and message storage into local databases. This enables the complete automation of the data exchange between participants and enables the enrichment of the model transformations with data-related aspects from the global (choreography) to the local (workflow) level. We support the authors arguments that the collaborating partners should specify the exchanged data and its structure in a commonly agreed global data model. This indeed strengthens the collaboration contract by ensuring correct data exchange in addition to correct conversations, i. e., order of messages. Instead of directly binding the data objects to databases on the level of the models, our approach introduces the TraDE middleware as an abstraction layer that provides capabilities to retrieve, cache and store data from different participants and data sources. This allows us to support other domains than the business domain, for example, in eScience data is commonly stored and exchanged through files in different formats. Moreover, unlike the approach from [30], we want to decouple the data flow from the message exchanges, without the need to exchange data only through the conversation messages. Our additional goal is to analyze and optimize the data transfer between the participants or services, so that as soon as the data are available and even before the (control flow) message that normally transports the data is sent, the data are transfered to the receiver's side and can be directly read when the corresponding message triggers the related operation of the invoked participant or service.

Barker et al. [4] define a new language for executable service choreographies called *Multiagent Protocols* (MAP) and introduce an open source framework for the enactment of MAP choreographies through the application of so-called *peers*. A peer provides extra functionality through a *choreography interface* that enables web services to participate in a choreography without requiring to adapt the underlying service implementations. One difference between the approaches is the use of standard versus non-standard notations and languages for choreography specifications which influences the portability of the models across modeling and execution environments. The same work presents an optimization of the data transfer between services, which in our view is a result of the paradigm shift between orchestration and choreography. Consequently, the demonstrated improvement results from the fact that the data is exchanged directly between choreographed services instead of being transferred through a centralized workflow engine. In our work, the focus is on optimizing the direct data exchange between the choreography participants during run time, no matter if they are implemented through workflows or services, by incorporating monitoring data and results from data dependence analysis to identify if, when and where data is actually required as described in Section II.

In [13], Barker et al. introduce the so-called *Circulate approach* that combines the advantages of both paradigms: orchestration and choreography. While the control flow remains orchestration-based, the data flow is conducted in a choreography-based manner. In order to enable services to transfer data between each other, *proxies* are introduced to provide the required functionality. Therefore, a proxy acts as an intermediary between the workflow engine and the actual services during service invocations. Consequently, the workflow engine triggers the invocation of services and the exchange of data between services through a proxy and the actual service invocations are performed by the proxies. We are following a similar approach by introducing the TraDE middleware to decouple the control flow from the data flow. In contrast to the Circulate approach where the data transfer between the proxies is controlled by the workflow engine and is therefore explicitly specified in the workflow models, we propose to annotate the workflow models and use the annotations in combination with the analyzed data dependencies to express the overall choreography data flow model, which can in turn be enacted by the TraDE middleware and serve as data flow coordinator (see Section II). As a result, (a) the workflow models are enriched instead of changed thus preserving the portability of the models on run time environments without TraDE support, and (b) outsourcing the control of the data flow to the TraDE middleware instead of explicitly specifying it through the workflow models provides more flexibility and higher optimization potential during run time since the data can flow independently of the control flow of the workflows or the choreography. For example, data can be transfered as soon as it is available to services invoked later while the workflow engine can continue navigation. Furthermore, data management tasks like data transformation, aggregation or split can be incorporated into the optimization to address different goals and requirements.

In [14], the authors introduce a *Flow-based Infrastructure for Composing Autonomous Services* (FICAS) and the *Compositional Language for Autonomous Services* (CLAS) used to express the relationships between collaborating services in a service composition. The resulting CLAS programs can then be executed by the FICAS runtime. Based on the analysis of data dependencies between the services, a CLAS program is decomposed into services and an execution plan for their composition. A central coordinator carries out the execution plan and thus controls the control and data flow and sends the corresponding commands to the services. The decoupling of control commands and data flow happens only on the level of the invoked services through separate data and control queues associated to each service for this purpose. The approach allows for a decentralized data exchange among services. The similarities to our approach are only in the notion of decentralized data exchange among services in a composition.

Binder et al. [11] introduce the concept of *Service Invocation Triggers* in order to allow for conducting service compositions in a decentralized manner. Such a trigger acts as a proxy for a specific service and collects all input data, invokes the service and routes the output data to successor triggers. In sum, the triggers deal with the data exchange and its ordering on behalf of the services in a service composition. It is our understanding that there is no global workflow model that is decomposed into triggers. The triggers themselves, however, contain partial knowledge of the workflow logic and the data dependencies among services to be invoked.

IV. Conclusion and Future Work

The importance of data-awareness and its effect on the BPM domain is increasing with the advances in the fields of eScience, Big Data or IoT. A more prominent role of data in service compositions is therefore a must in order to benefit from these developments. Existing research showed that introducing data on the level of service choreographies, as a means to specify a global collaboration contract between interacting parties, provides valuable benefits for both modeling and execution aspects. In this paper, we introduced a management life cycle of service choreographies that natively supports data-awareness through the so-called TraDE methods for data flow analysis, optimization and transparent data exchange. Since most of the existing approaches only concentrate on modeling or execution aspects, our goal is to provide an end-to-end approach for data-aware service choreographies that support data-related aspects throughout all life cycle phases. In particular, while in terms of modeling further steps from the global (choreography) to the local (workflow) models can be automated and enriched to reduce data-related refinement efforts, in terms of execution the knowledge about the data allows to decouple the data flow from the control flow and handle it in a decentralized manner by exchanging data directly between the composed services as well as further optimizations of the data exchange during run time.

In our future work, we will define formal models for the introduced data-related artifacts of the TraDE methods, namely Choreography Data Model, Choreography Data Dependence Graph and Participant Data Dependence Graph, as a foundation for the definition of corresponding algorithms that realize the required generation, analysis, transformation and optimization functionality. The resulting algorithms and formal models will be implemented and integrated into our existing choreography and workflow modeling environment to support the modeling, transformation and refinement of data-aware service choreographies. We will also update our prototypical realization and provide evaluation based on real world scenarios.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart.

References

[1] C. Peltz, "Web services orchestration and choreography," *Computer*, vol. 36, no. 10, pp. 46–52, Oct. 2003.

[2] G. Decker, O. Kopp, and A. Barros, "An introduction to service choreographies," *Information Technology*, vol. 50, no. 2, pp. 122–127, 2008.

[3] A. Barros, M. Dumas, and P. Oaks, "A critical overview of the web services choreography description language," *BPTrends Newsletter*, vol. 3, pp. 1–24, 2005.

[4] A. Barker, C. Walton, and D. Robertson, "Choreographing web services," *Services Computing, IEEE Transactions on*, vol. 2, no. 2, pp. 152–166, Apr. 2009.

[5] A. Weiß and D. Karastoyanova, "Enabling coupled multi-scale, multi-field experiments through choreographies of data-driven scientific simulations," *Computing*, pp. 1–29, 2014.

[6] K. Görlach, M. Sonntag, D. Karastoyanova, F. Leymann, and M. Reiter, *Guide to e-Science: Next Generation Scientific Research and Discovery*. Springer London, 2011, ch. Conventional Workflow Technology for Scientific Simulation, pp. 323–352.

[7] R. Barga and D. Gannon, *Workflows for e-Science: Scientific Workflows for Grids*. Springer London, 2007, ch. Scientific versus Business Workflows, pp. 9–16.

[8] A. Slominski, *Workflows for e-Science: Scientific Workflows for Grids*. Springer London, 2007, ch. Adapting BPEL to Scientific Workflows, pp. 208–226.

[9] R. Schmidt, M. Möhring, S. Maier, J. Pietsch, and R.-C. Härting, "Big data as strategic enabler - insights from central European enterprises," in *Business Information Systems*, ser. Lecture Notes in Business Information Processing. Springer International Publishing, 2014, vol. 176, pp. 50–60.

[10] S. Meyer, K. Sperner, C. Magerkurth, and J. Pasquier, "Towards modeling real-world aware business processes," in *Proceedings of WoT'11*. ACM, 2011, pp. 8:1–8:6.

[11] W. Binder, I. Constantinescu, and B. Faltings, "Decentralized orchestration of composite web services," in *Proceedings of ICWS'06*, Sep. 2006, pp. 869–876.

[12] A. Barker, J. B. Weissman, and J. Van Hemert, "Eliminating the middleman: Peer-to-peer dataflow," in *Proceedings of HPDC'08*. ACM, 2008, pp. 55–64.

[13] A. Barker, J. B. Weissman, J. Van Hemert *et al.*, "Reducing data transfer in service-oriented architectures: The circulate approach," *Services Computing, IEEE Transactions on*, vol. 5, no. 3, pp. 437–449, 2012.

[14] D. Liu, K. H. Law, and G. Wiederhold, "Data-flow distribution in ficas service composition infrastructure," in *Proceedings of ICPDCS'02*. Citeseer, 2002.

[15] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer Science & Business Media, 2012.

[16] A. Weiß and D. Karastoyanova, "A life cycle for coupled multi-scale, multi-field experiments realized through choreographies," in *Proceedings of EDOC'14*, Sep. 2014, pp. 234–241.

[17] M. Hahn, D. Karastoyanova, and F. Leymann, "Data-Aware Service Choreographies through Transparent Data Exchange," in *Proceedings of ICWE'16*, ser. LNCS. Springer, 2016.

[18] A. Weiß, V. Andrikopoulos, M. Hahn, and D. Karastoyanova, "Fostering reuse in choreography modeling through choreography fragments," in *Proceedings of ICEIS'15*. SciTePress, Apr. 2015, pp. 28–36.

[19] F. Leymann and D. Roller, *Production Workflow - Concepts and Techniques*. PTR Prentice Hall, Jan. 2000.

[20] G. Decker, O. Kopp, F. Leymann, and M. Weske, "BPEL4Chor: Extending BPEL for modeling choreographies," in *Proceedings of ICWS'07*. IEEE, 2007.

[21] Object Management Group (OMG), "Business process model and notation (BPMN) version 2.0," Object Management Group, Inc, Jan. 2011.

[22] J. Zaha, A. Barros, M. Dumas, and A. ter Hofstede, "Let's dance: A language for service behavior modeling," in *OTM 2006 Conferences*, ser. LNCS. Springer Berlin Heidelberg, 2006, vol. 4275, pp. 145–162.

[23] Organization for the Advancement of Structured Information Standards (OASIS), "Web services business process execution language version 2.0," OASIS Standard, 2007.

[24] P. Reimann, "Generating BPEL processes from a BPEL4Chor description," Student Thesis No. 2100, p. 107, 2007.

[25] D. J. Kuck, R. H. Kuhn, D. A. Padua, B. Leasure, and M. Wolfe, "Dependence graphs and compiler optimizations," in *Proceedings of POPL'81*. ACM, 1981, pp. 207–218.

[26] K. J. Ottenstein, "Data-flow graphs as an intermediate program form," Ph.D. dissertation, 1978.

[27] A. Weiß, V. Andrikopoulos, M. Hahn, and D. Karastoyanova, "Rewinding and repeating scientific choreographies," in *OTM 2015 Conferences*, ser. LNCS. Springer International Publishing, 2015, vol. 9415, pp. 337–347.

[28] B. Wetzstein, D. Karastoyanova, O. Kopp, F. Leymann, and D. Zwink, "Cross-organizational process monitoring based on service choreographies," in *Proceedings of SAC'10*. ACM, 2010, pp. 2485–2490.

[29] J. Koehler and J. Vanhatalo, "Process anti-patterns: How to avoid the common traps of business process modeling," *IBM WebSphere Developer Technical Journal*, vol. 10, no. 2, p. 4, 2007.

[30] A. Meyer, L. Pufahl, K. Batoulis, D. Fahland, and M. Weske, "Automating data exchange in process choreographies," *Information Systems*, 2015.

All links were last followed on May 12, 2016.