Institute of Architecture of Application Systems

Fostering Reuse in Choreography Modeling Through Choreography Fragments

Andreas Weiß, Vasilios Andrikopoulos, Michael Hahn, Dimka Karastoyanova

Institute of Architecture of Application Systems, University of Stuttgart, Germany,
firstname.lastname@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{INPROC-2015-07,
  author    = {Andreas Wei{\ss} and Vasilios Andrikopoulos and Michael Hahn and
               Dimka Karastoyanova},
  title     = {{Fostering Reuse in Choreography Modeling Through Choreography
               Fragments}},
  booktitle = {Proceedings of the 17th International Conference on Enterprise
               Information Systems ICEIS 2015},
  year      = {2015},
  pages     = {28-36},
  doi       = {10.5220/0005342000280036},
  publisher = {SciTePress}
}
```

These publication and contributions were presented at ICEIS 2015
ICEIS 2015 Web site: http://www.iceis.org/

© 2015 SciTePress. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the SciTePress.

Universität Stuttgart
Germany

Fostering Reuse in Choreography Modeling Through Choreography Fragments

Andreas Weiß, Vasilios Andrikopoulos, Michael Hahn, and Dimka Karastoyanova
Institute of Architecture of Application Systems (IAAS)
University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
{*andreas.weiss, vasilios.andrikopoulos, michael.hahn, dimka.karastoyanova*}@*iaas.uni-stuttgart.de*

Keywords: Choreography, Choreography Fragment, Choreography Fragment Pattern, Choreography Modeling, Reuse

Abstract: The concept of reuse in process models is extensively studied in the literature. Sub-processes, process templates, process variants, and process reference models are employed as reusable elements for process modeling. Additionally, the notion of process fragments has been introduced to capture parts of a process model and store them for later reuse. In contrast, concepts for reuse of processes that cross the boundaries of organizations, i.e., choreographies, have not yet been studied in the appropriate level of detail. In this paper, we introduce the concept of choreography fragments as reusable elements for choreography modeling. Choreography fragments can be extracted from choreography models, adapted, stored, and inserted into new models. We provide a formal model for choreography fragments and identify a set of patterns constituting frequently occurring meaningful choreography fragments.

1 INTRODUCTION

Reuse on the level of the modeling of executable business processes, also known as workflows or service orchestrations, is a well documented area of research that has introduced many interesting concepts and techniques. For example, frequently used orchestration logic can be sourced to sub-processes during modeling time to be invoked later from parent process models (Leymann and Roller, 2000). Process templates, process variants, and process reference models (Gottschalk et al., 2007) are employed as reusable elements for process modeling. Furthermore, the notion of *process fragments* has already been introduced to capture parts of a process model and store them for later reuse (Schumm et al., 2011). However, in contrast to individual workflow models, concepts for reuse of workflows that cross the boundaries of organizations, so-called *choreographies*, have not yet been studied in sufficient detail. Choreographies model the interconnection of independent business organizations or units by showing only the publicly visible communication behavior of the choreography participants. The participants are implemented by services or orchestrations of services, i.e., workflows. Choreographies can be described with two modeling approaches: interaction models, or interconnected interface behavior models (Decker et al., 2008a). For the purposes of this work, we refer to interconnected interface behavior models when using the term choreography model.

In this paper we introduce the notion of *choreography fragments* in order to provide the means for reuse in choreography modeling. Choreography fragments can be used to capture and extract modeling best practices and recurring patterns in choreography models, store them in a library, and insert them into new choreography models to foster reuse for choreography modeling. Furthermore, and in particular for partially defined choreography models with abstract constructs as discussed in (Weiß et al., 2014), choreography fragments can also be used for the refinement of the abstract constructs into concrete choreography model elements. Similar to process fragments, choreography fragments can be created in either a top-down or bottom-up manner (Schumm et al., 2010). Top-down means the creation of fragments through extraction from choreography models or mining of audit trails of interconnected workflows, whereas bottom-up refers to the manual creation of choreography fragments from scratch. In this work, we concentrate on the top-down approach, based on a formal model that abstracts from the underlying modeling language used.

An additional benefit of the use of choreography fragments arises by looking across different types

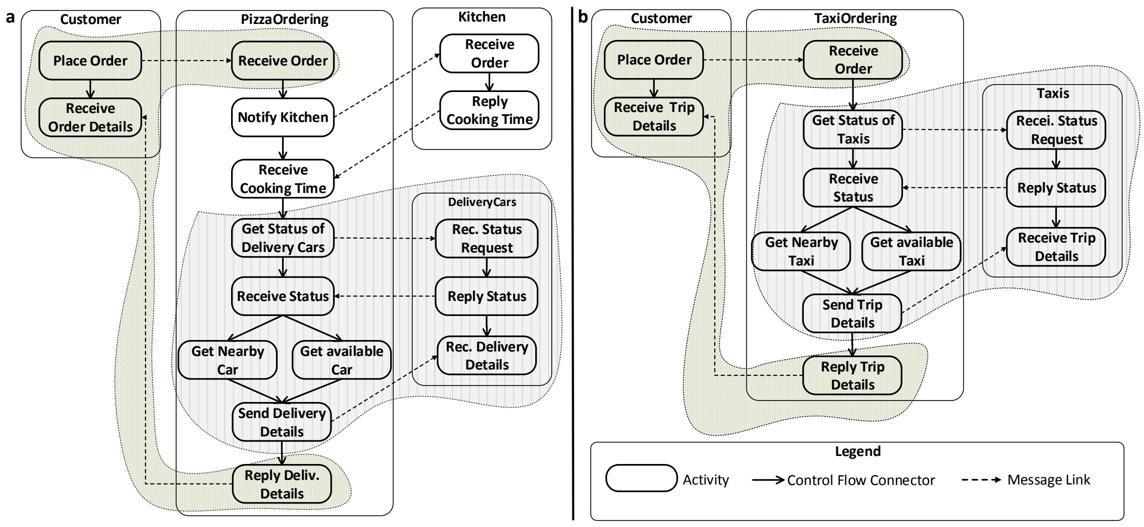

Figure 1: Motivating scenario: (a) Pizza Company. (b) Taxi Company. Adopted from (Schilling, 2014)

of choreography fragments from different domains. This allows the identification of reusable *choreography fragment patterns* that appear frequently during the extraction of fragments in choreographies. These patterns help identifying meaningful choreography fragments and thus further promote the concept of reusability in modeling.

The contribution of this work is therefore twofold. On the one hand, we provide a *formal model for choreography fragments* that is independent of a particular choreography modeling language and therefore reusable across technologies. On the other hand, based on this formal model we discuss an (initial) list of *fragment patterns* that represent commonly appearing constructs in choreography fragments.

The rest of this paper is structured as follows. We first show a motivating scenario (Sec. 2), and then provide a formal model defining the concept of choreography fragments (Sec. 3). In Sec. 4, we identify choreography fragment patterns that constitute meaningful choreography fragments. An evaluation of the concepts is conducted by analyzing choreography models from literature (Sec. 5). Sec. 6 compares our work to related ones. Finally, we conclude our paper with a summary and an outlook to future work in Sec. 7.

2 MOTIVATION

In this section, we introduce a scenario to motivate the concept of choreography fragments. The example in Fig. 1 is based on the one from (Schilling, 2014). Fig 1a shows the interconnected workflows of a pizza company forming a choreography. A customer can order a pizza using a smartphone app. The PizzaOrdering participant notifies the kitchen system about the incoming order and requests the status of the delivery personnel and their cars via a process on a mobile device. Depending on the statuses of the delivery personnel either the nearest or any available car is chosen to deliver the pizza. The customer and the chosen car are subsequently notified about the delivery details. Let's assume now that the pizza company discovers that their delivery cars can also be used to transport customers in a shared economy service model. Its IT department or an external consultant now has to model and implement the relevant business processes rapidly to seize this business opportunity before the competition. When looking at the new taxi company choreography in Fig. 1b, it becomes apparent that the overall structure of the choreography is similar to the pizza company choreography. If a human modeler had the means to extract the similarities from the existing pizza company choreography such as the communication between the customer process and the order process, store them as reusable fragments, and insert them when modeling a new choreography spanning several systems, overall time-to-market would decrease and choreography models would no longer need to be created from scratch. Therefore, even in small-scale choreography models like the ones discussed in Fig. 1 there exists a potential for reusability of modeling constructs that the rest of this work builds upon.

3 FORMAL MODEL

Our notion of executable process models is oriented on the work of (Sonntag and Karastoyanova, 2012), which is itself based on the formal model introduced in (Leymann and Roller, 2000). Our definition of process fragments adopts the formal model of (Eberle et al., 2010). Subsequently, we extend these definitions to introduce the concepts of choreography models and fragments. A choreography model consists of at least two *participants*, i.e., process models, *message links*, and explicit *control flow* and implicit *data flow* via the manipulation of *variables*. Choreography models typically only show the publicly visible communication behavior, because the details of the workflows implementing the choreography participants are considered sensitive information providing a competitive advantage to business organizations. The usually non-executable choreography models are defined collaboratively and used to generate abstract representations of the choreography participants in a workflow language. The collaborating business organizations then refine their workflow models with business logic. In addition to control and data flow for communication purposes, our definition of choreography model and choreography fragment also allows for the existence of business logic related activities, variables, and links. This allows us to use process fragments as elements of reuse in the context of choreographies.

Definition 1 formally specifies our understanding of process models, which are then used as choreography participants in a choreography model. Note that we use the projection operator π_n to access the n^{th} element of a tuple starting from index 1. $\mathcal{P}(X)$ denotes the power set of the set X including the empty set $\emptyset$. $p_y.X$ accesses element X (potentially a set) of element p_y.

Definition 1 (Process Model, G)**.** A *process model* is a directed acyclic graph represented by the tuple $G = (m, V, i, o, A, L)$, where $m \in M$ is the name of the process model, $V \subseteq M \times S$ (M = set of names; S = set of data structures) is the set of *variables*, i is the *map of input variables*, o is the *map of output variables*, A is the set of *activities*, and L is the set of *control connectors* (control flow links). Input variables providing data to activities can be assigned using an input variable map $i : A \rightarrow \mathcal{P}(V)$. Output variables to which activities may write data to are described by the output variable map $o : A \rightarrow \mathcal{P}(V)$. Finally, the set of control connectors is $L \subseteq A \times A \times C$. A link $l \in L$ is a triple $l = (a_{source}, a_{target}, t \mid a_{source}, a_{target} \in A, t \in C \wedge a_{source} \neq a_{target})$ connecting a source and a target activity, while its *transition condition* (where C is the set of all conditions) is evaluated during run time. An evaluation to true means that the link is followed. An activity $a_{start} \in A$ is called *start activity* if it is not the target of a control flow link: $A_{start} \subseteq A := \{a \mid a \in A \wedge \forall l \in L, a \neq \pi_2(l)\}$. The set G_{all} is the set of all process model graphs.

Definition 2. (Process Fragment, F). A *process fragment* is a directed, acyclic graph $F \leq_F G$ represented by a tuple $F = (m, V, i, o, A, L)$. The $\leq_F$ operator means, that the components of the tuple F are a subset or equal to the their corresponding components in the process model tuple G. The definition of a control connector $l \in L$ is extended by the concept of *dangling control connectors*. A dangling control connector is a triple $l = (\perp, a_{target}, t)$ or $l = (a_{source}, \perp, t)$, where $\perp$ is a missing (undefined) source or target activity. Therefore, the set L of control connectors is now defined as $L \subseteq (A \cup \perp) \times (A \cup \perp) \times C$, where C is the set of transition conditions (see Definition 1).

In the following, Definitions 1 and 2 are used as a basis for the definitions of choreography model and choreography fragment. We adopt the concept of typed participants and so-called participant sets from (Decker et al., 2007):

Definition 3. (Choreography Model, $\mathfrak{C}$). A choreography model is a directed, acyclic graph denoted by the tuple $\mathfrak{C} = (m, P, P_{set}, ML)$, where $m \in M$ is the name of the choreography model, P is the set of choreography participants, P_{set} is the set containing participant sets, ML is the set containing all message links between the choreography participants. A *choreography participant* $p \in P$ is a triple $p = (m, type, G)$, where $m \in M$ is the name of the participant, $type : P \rightarrow T$ is the function assigning a type $t_p \in T$ to the participant, and G is a process model graph as defined in Definition 1. Typing the participant allows to express if several participants of the same type participate in the same choreography. The set of all participants is denoted by P_{all}. A *participant set* $p_{set} \in P_{set}$ is described by $p_{set} \subseteq P_{all}$. This modeling construct is used to model a set of choreography participants whose number can be determined only during execution.We define a *containment relation* R_{con} as $R_{con} : P_{set} \rightarrow \mathcal{P}(P_{all})$. R_{con} indicates the participants contained in a participant set p_{set}. Contained participants in a participant set must be of the same type and have the same process model graph: $\forall p_x, p_y \in R_{con}(p_{set}) : type(p_x) = type(p_y) \wedge p_x.G = p_y.G$.

The set of *message links* ML is denoted as $ML \subseteq (P \cup P_{set}) \times P \times A \times A \times C$. A message link is a tuple $ml \in ML = (p_s, p_r, a_s, a_r, t)$, where p_s, p_r are the sending and receiving participants. Furthermore, p_s can be a participant set if any contained participant p

in p_s: $p \in R_{con}(p_s)$, $p_s \in \pi_3(\mathfrak{C})$ may send something. For the sending and receiving participants the following holds: $p_s \neq p_r$, i.e., the sender and the receiver must not be identical; $a_s \in \pi_5(p_s.G)$ and $a_r \in \pi_5(p_r.G)$ are the sending and receiving activities for which the following holds: $a_s \neq a_r$. Sending and receiving activities must have only one outgoing or incoming message link. These activities are denoted as *communication activities*. The transition condition $t \in C$ is evaluated during run time. Choreography participants and participant sets are only connected via message links. However, a choreography model graph may have disconnected components because there may be participants or participant sets that are not connected to other participants or participant sets via message links. Note that the disconnection is only allowed between participants but not between activities of a participant's process graph.

The defined concepts can be illustrated by means of the motivating example in Fig. 1. The participants are the workflows *Customer*, *Pizza/TaxiOrdering*, *Kitchen*, and *DeliveryCar/Taxi*. Message links exist between the participants. An example is the message link between the *Customer* and the *PizzaOrdering* workflow. *Customer* is the sending participant and *Place Order* the sending activity of the message link. *PizzaOrdering* is the receiving participant and *Receive Order* the receiving activity. The *DeliveryCar/Taxi* workflow can be modeled as participant set as the number of instances may be only known at run time. However, the type of the instances will always be the same.

Definition 4. (Choreography Fragment, $\mathfrak{F}_c$). A choreography fragment $\mathfrak{F}_c \preceq_{\mathfrak{F}_c} \mathfrak{C}$ is a directed, acyclic graph with possibly disconnected components and is represented by a tuple $\mathfrak{F}_c = (m, P_f, P_{set}, ML)$. That means that the choreography fragment tuple components are a subset or equal to the corresponding choreography model tuple components. A participant $p_f \in P_f$ is defined differently compared to the participant in a choreography model: $p_f = (m, type, F)$, where F is a process fragment. That means, in a choreography fragment a participant contains a process fragment, which can also be a process model (cf. Definition 2). The definition of a message link $ml \in ML$ is extended with the concept of dangling message links. A *dangling message link* is a tuple $ml = (\perp, p_r, \perp, a_r, t)$ or $ml = (p_s, \perp, a_s, \perp, t)$, where $\perp$ is a missing sending participant and sending activity or a missing receiving participant and receiving activity. The set of message links ML in a choreography fragment $\mathfrak{F}_c$ is defined as $ML \subseteq (P \cup P_{set} \perp) \times (P \cup \perp) \times (A \cup \perp) \times (A \cup \perp) \times C$, where C is the set of (transition) conditions. Distinct participants may be disconnected, i.e., without message links to or from any other participant in the choreography fragment. Depending on the selection of elements that should belong to a choreography fragment, activities inside participants may become disconnected. To represent a *valid* choreography fragment these disconnections must be repaired, either manually or automatically. The only exception are activities from parallel paths that were not connected in the original model.

As an example, the choreography fragment involving the Customer and the PizzaOrdering participant in Fig. 1a can be expressed using our formal model in the following way: $\mathfrak{F}_c = \{$*CustomerOderingFragment,* $\{$*Customer, PizzaOrdering*$\}, \emptyset, \{ml_1, ml_2\}\}$. The set of participant sets is empty in this example. The participant *Customer* is the tuple *Customer* $= \{$*Customer, customerType,* $customer_{process}\}$ and the participant *PizzaOrdering* is the tuple $\{$*PizzaOrdering, pizzaOrderingType,* $pizzaOrdering_{process}\}$. The process graph $customer_{process}$ is the tuple $\{CustomerProcess, \emptyset, \emptyset, \emptyset, \{$*PlaceOrder, ReceiveOrderDetails*$\}$, $\{$*PlaceOrder, ReceiveOrderDetails, true*$\}\}$. The process graph $pizzaOrdering_{process}$ is the tuple $\{$*PizzaOrderingProcess,* $\emptyset, \emptyset, \emptyset$, $\{$*ReceiveOrder, ReplyDeliveryDetails*$\}$, $\{$*ReceiveOrder, ReplyDeliveryDetails, true*$\}\}$. Since we have not introduced variables in our example in Fig. 1, the corresponding sets are empty. The control flow links in each of the two process graphs have the condition true by default. In the $pizzaOrdering_{process}$ process graph the included activities have been reconnected with a control flow link in order to obtain a valid choreography fragment. The message links ml_1 and ml_2 contain the following elements: $ml_1 = \{$*Customer, PizzaOrdering, PlaceOrder, ReceiveOrder, true*$\}$, $ml_2 = \{$*PizzaOrdering, Customer, ReplyDeliveryDetails, ReceiveOrderDetails, true*$\}$.

Our formal model expresses data flow via implicit data connectors, i.e., the manipulations of variables by activities. When creating a process/choreography fragment, the "dangling" data references must also be included into the fragment. These are all variables an activity inside a participant reads from or writes to. Therefore, all participants from the sets of participants have to be considered: $\forall p \in \pi_2(\mathfrak{F}_c), \forall a \in \pi_5(p.F) : V_f \in \pi_2(p.F) \subseteq \{v \mid i(a) \cup o(a)\}$. The same holds for all the participants contained in participant sets: $\forall p_{set} \in \pi_3(\mathfrak{F}_c), \forall p \in R_{con}(p_{set}), \forall a \in \pi_5(p.F) :$ $V_f \in \pi_2(p.F) \subseteq \{v \mid i(a) \cup o(a)\}$.

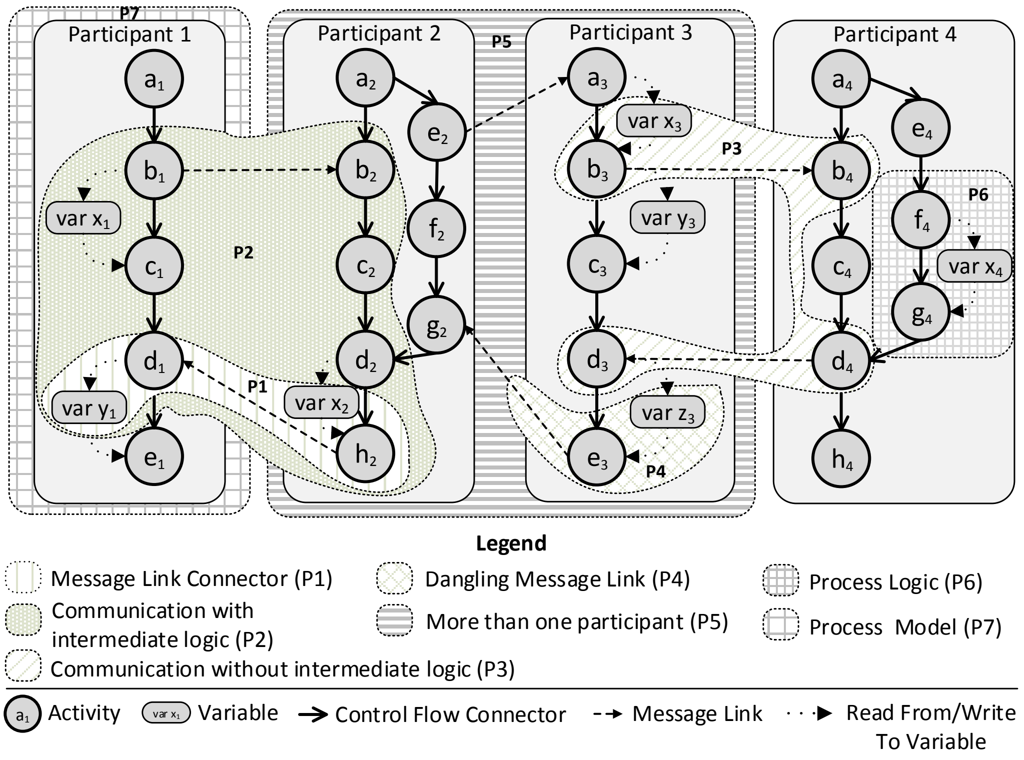

Figure 2: Examples of the choreography fragment patterns

4 CHOREOGRAPHY FRAGMENT PATTERNS

Previous work (Andrikopoulos et al., 2014) has introduced the notion of Collaborative, Dynamic & Complex (CDC) systems that provide unified concepts and tools for different application domains with overlapping requirements but so far divergent solution. CDC systems propose to model and enact problems of application domains such as scientific experiments (Weiß and Karastoyanova, 2014) and Collective Adaptive Systems (CAS) (Andrikopoulos et al., 2013) as choreographies. When working with choreography models from different application domains we observed that they share common elements that can be expressed by a *pattern*. Some of these patterns have already been described by the *service interaction patterns* of Barros et al. (Barros et al., 2005). However, service interaction patterns can be subsumed into more general choreography fragment patterns.

In the following, we discuss some choreography fragment patterns that we identified and their properties, while relating to the examples in Fig. 2. For the rest of the discussion we refer to the act of copying parts of a choreography model to other models as *extraction*. For this purpose, model elements in the original choreography model must be *selected* either manually with the help of a graphical tool, or automatically to indicate what has to be *copied*. The copying of model elements should always result again in a valid choreography model. Our patterns are meant to show human modelers which parts of a choreography model can be extracted to form a valid fragment that can be reused in another choreography model.

Message link connector (P1): A *message link connector* is a tuple $ml_c = (a_s, i(a_s), o(a_s), ml, a_r, i(a_r), o(a_r))$, where $ml \in ML$ and $a_s = \pi_3(ml)$, $a_r = \pi_4(ml)$ and $a_s, a_r \neq \perp$, with $i(a_s)$, $o(a_s)$, $i(a_r), o(a_r)$ the input and output variable maps of the activities a_s and a_r, respectively. This means that the sending activity in tuple ml_c is equal to the sending activity in the contained message link ml and the receiving activity in the tuple ml_c is equal to the receiving activity in the contained message link ml, and the message link ml must not be a dangling message link. Furthermore, the message link connector contains the input and output variables of the sending and receiving activities. Message link connectors can be used to ease the modeling of a connection between two participants, which typically involves the separate definition of a sending and receiving activity as well as a message link connecting both activities. In Fig. 2, the message link connector is represented by the activities h_2 and d_1 in the participants 2 and 1, respectively, and the message link from activity h_2 to d_1. Variables x_2 and y_1 are also part of the message link connector.

Communication with intermediate logic (P2): The message link connector described in P1 is a special case of a choreography fragment distributed among several choreography participants. The generalization of this pattern can have parts of the choreography logic distributed among several participants with message links between them. The logic inside the participants between the sending and receiving activities (communication activities b_1, d_1 and b_2, h_2 in Fig. 2) is completely selected. In our example, besides the already enumerated communication activities, the activities c_1 and c_2, d_2 as well as the message links $b_1 - b_2$ and $h_2 - d_1$ belong to this choreography fragment pattern. The cardinality of participants in the fragment is $|\pi_2(\mathfrak{F}_c)| > 1$ or the cardinality of participant sets in the fragment is $|\pi_3(\mathfrak{F}_c)| > 1$. Moreover, the participants must be connected by message links and there must be at least two message links in the choreography fragment: $\forall p_f \in \pi_2(\mathfrak{F}_c) : \exists ml \in \pi_4(\mathfrak{F}_c)$, where $p_f = \pi_1(ml) \vee p_f = \pi_2(ml)\}$. For participant sets the following holds: $\forall p_{set} \in \pi_3(\mathfrak{F}_c), \forall p_f \in R_{con}(p_{set}) :$ $\exists ml \in \pi_4(\mathfrak{F}_c)$, where $p_f = \pi_1(ml) \vee p_f = \pi_2(ml)$ or instead $p_{set} = \pi_1(ml)$. That means, for all participants contained in each participant set there exists a message link such that each participant is either a sending or receiving participant or the whole participant set is the sender of a message link instead. In any case, the cardinality of ML must be $|ML| \geq 2$. Furthermore, the number of activities in the participants of a choreography fragment has to be smaller than in the choreography model the fragment originates from. This is expressed using the *origin* function[1]: $\forall p_f \in \pi_2(\mathfrak{F}_c) : |\pi_5(p_f.F)| < |\pi_5(origin_P(p_f, \mathfrak{C}).G)|$. This also holds for the participants contained in each participant set. P2 subsumes the service interaction patterns (Barros et al., 2005) and message exchange patterns (MEP). That means, choreography modelers can extract choreography fragments that represent these known patterns from literature and reuse them in a new choreography model.

Communication without intermediate logic (P3): This pattern is a restriction of P2, and also subsumes the service interaction patterns (Barros et al., 2005) from literature. In contrast to P2, here (domain-specific) model elements including variables between the communication activities realizing these patterns are completely omitted when creating a new choreography fragment. In our example, the activities c_3 and c_4 and the variable y_3 are not part of P3. Input variables to affected communication activities, however, may be included in the fragment (here variable x_3). Note that the communication activities such as send and receive are disconnected after extraction. The extraction algorithm should try to preserve this connection by providing re-connection suggestions to the modeler extracting the fragment or by placing abstract activities (Weiß et al., 2014) in between. The formalization presented for P2 is extended by the (temporary) disconnection between the communication activities when extracting a choreography fragment described by this pattern: $\exists a \in \pi_5(p_f.F), p_f \in \pi_2(\mathfrak{F}_c) \vee p_f \in R_{con}(p_{set}), p_{set} \in \pi_3(\mathfrak{F}_c)$, for which $\nexists l \in \pi_6(p_f)$, where $a = \pi_2(l)$ and additionally: $origin_A(a, \mathfrak{C}) \notin A_{start}$. That means, there exists a disconnection if the activities in question are not target of a control connector and were not a start activity in the original choreography model. The origin function here returns the activity of the original choreography model for an activity of a choreography fragment.

Dangling message link (P4): In this pattern, a message link $ml \in \pi_4(\mathfrak{F}_c)$ is not connected to activities and participants on both ends but rather it is dangling on one side (cf. Definition 4). The non-dangling side of the message link is connected to other elements in a choreography fragment.

More than one choreography participant (P5): A choreography fragment can comprise several participants and/or participant sets of a choreography model. Ultimately, the choreography fragment can contain a completely specified choreography model. This pattern is an extension to P2 (communication with intermediate logic), however, here the complete choreography participants are attributed to a choreography fragment. The formalization is identical to P2, except that the constraint on the number of activities in the participants of the choreography fragment is removed.

Process logic (P6): The logic inside a participant can be seen as a process fragment as specified in Definition 2. This logic can be extracted as a choreography fragment. This includes control and data flow via variable manipulation as well as dangling control connectors.

Process model (P7): A complete process model as defined in Definition 1 can be a choreography

[1]The origin function is defined as $origin_P : P_f \times \mathfrak{C} \rightarrow P$.

fragment. Inserting a process model into a choreography model converts it into a choreography participant.

These patterns represent a (not necessarily complete) list of reusable fragment patterns that can be used for the extraction and further insertion of choreography fragments from and to choreography models, respectively. In the next section, the identified patterns will be evaluated by analyzing existing choreography models from literature.

5 EVALUATION

For purposes of evaluating our proposal, we analyzed a set of choreography models from the literature that cover different domains from business scenarios to scientific experiments. Table 1 shows how often choreography fragments described by a particular pattern occurred in a particular choreography model. Every choreography model naturally has some message links and corresponding sending and receiving communication activities. Thus, for each choreography model a set of message link connectors can be identified (P1). The number of identified message link connectors typically corresponds to the number of message links in the choreography model. However, some models contain black box participants that do not reveal any internal logic, thus, the number of message link connectors is smaller than the number of message links for those choreography models. The number of dangling message links (P5) typically is twice the number of message links in the analyzed choreography model, with the exception of the models containing black box participants. The communication with intermediate logic pattern (P2) also occurs in almost every choreography model we analyzed. An exception is the scientific choreography, which contains a strictly sequential invocation between the two participants, thus, P2 is not observed. For each analyzed choreography model, P3 contains the same number of occurrences as P2. This is due to the fact that P3 is a restriction of P2 that omits the logic between the communication activities, but specifies identical service interaction patterns. Again, the scientific choreography is an exception here. P5 shows the number combinations of complete participants and the message links between them that can be included in a choreography fragment for each of the analyzed models. The number of process models (P7) corresponds to the number of participants in a choreography model. The counting of the possible 2^{n-1} process fragments (P6) per choreography participant, where $n = |\pi_5(p.G)|$ is the number of activities in choreography participant, is out of scope of this paper. However, it is still a valid choreography fragment pattern because it can be combined with the other patterns to form more complex choreography fragments.

The evaluation shows that all choreography fragment patterns introduced in Sec. 4 can be found in the analyzed choreography models in a sufficient number. However, it is clear that this initial evaluation does not provide any evidence towards the completeness of this list. More comprehensive evaluations are necessary to clarify this. Furthermore, the occurrence of a pattern depends on the structure of the choreography model. For example, the choreography model for scientific experiments is sequential in nature due to domain-specific requirements, thus, choreography fragments implementing P2 and P3 can not be found in this model.

6 RELATED WORK

The concept of process fragments as an element of reuse in developing process based applications has been covered in several works (Markovic and Pereira, 2008; Weber et al., 2008; Eberle et al., 2010; Schumm et al., 2011). For example, Markovic and Pereira (Markovic and Pereira, 2008) introduce a π-calculus based approach for process fragments. Process fragments can be semantically annotated for querying or autocompletion of process models. In (Weber et al., 2008), the concept of process fragments is used to describe change patterns and change reuse for process-aware information systems. However, these works lack a discussion of modeling elements that only exist on the level of choreographies such as participant sets or explicit message links connecting participants. This gap is closed with our notion of choreography fragments. Eberle et al. (Eberle et al., 2010) introduce a formal model for process fragments and corresponding composition operations. We have adopted the formal model for process fragments and extended it to also consider choreography fragments. In (Schumm et al., 2011), the authors propose the concept of a process fragment choreography by using process fragments to describe a collaboration scenario. This concept corresponds to the pattern described in P2.

The WS-CDL choreography language (Kavantzas et al., 2005) has the Perform Activity construct that allows the recursive composition of existing choreographies into more complex choreographies. While the composition is also a method of reuse, our definition of choreography fragments not only comprises completely specified choreography models but also

	Patterns						
Choreography Models	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
Pizza Company (Section 2)	7	5	5	14	5	*	4
Taxi Company (Section 2)	5	3	3	10	3	*	3
Travel Scenario (Decker et al., 2007)	7	8	8	14	4	*	3
Auctioning Scenario (Decker et al., 2009)	11	8	8	22	4	*	3
Taxi Company (Andrikopoulos et al., 2014)	5	4	4	10	4	*	4
Logistics Scenario (Object Management Group, 2011b)	13	11	11	26	6	*	4
Scientific Choreography (Weiß and Karastoyanova, 2014)	1	0	0	2	2	*	2
Book Purchase (Weske, 2012)	5	5	5	10	3	*	3
Incident Management I (Object Management Group, 2011a)	6	6	6	16	5	*	4
Incident Management II (Object Management Group, 2011a)	8	8	8	20	8	*	5
Nobel Price Process (Object Management Group, 2011a)	5	3	3	11	7	*	4
Delivery Scenario (Decker et al., 2008b)	6	6	6	12	3	*	3
Car Development (Wagner et al., 2011)	4	3	3	8	1	*	2

Table 1: Number of pattern occurrences in the analyzed choreography models

parts of a choreography model such as a message link connector to ease modeling for human users. Furthermore, our concept of choreography fragments is not tied to a specific choreography modeling language. Reuse of choreography models is also proposed by (Eder et al., 2006), who introduce federated choreographies, which are shared between partners and may support other choreographies. The choreographies are realized by private orchestrations possibly contributing to more than one choreography. The main difference to our work is the specification of choreographies with interaction models.

So-called partial choreographies allowing the definition of choreographies where the implementation of roles can be left open until run time are proposed in (Montesi and Yoshida, 2013). At run time, the partial choreographies are composed. The authors aim for the programming of choreographies using calculus, while we want to support modeling with reusable elements extracted from choreography models.

The capabilities of choreography languages can be compared using the service interaction patterns (Barros et al., 2005). The service interaction patterns include for example the *send/receive* or *one-to-many send* pattern. Our work builds on these patterns and subsumes them into more general ones. Moreover, when analyzing choreography models additional patterns constituting meaningful choreography fragments can be identified. These patterns can be used to support the work of choreography modelers. Our patterns in conjunction with the service interaction patterns can be used to select meaningful choreographies fragments for extraction.

7 CONCLUSIONS AND FUTURE WORK

In this work, we have introduced the concept of choreography fragments as elements of reuse in choreography modeling. Based on the formal definitions of process models and fragments we have provided a formal model for choreography models and fragments. Choreography fragments represent parts of choreographies that can be reused across different choreography models in different domains. Based on this formalization and on existing work we identified a set of fragment patterns that constitute frequently appearing fragment constructs that can be used as the basis for fragment extraction in choreography models.

Our next step in the context of choreography fragments is the definition of a method to support human choreography modelers with the (semi-)automatic extraction of choreography fragments from choreography models, the storing the fragments in a fragment library, and the insertion of choreography fragments into a new or existing choreography model. This includes the definition of algorithms that support the extraction of selected elements from a choreography model based on user input and the creation of valid choreography fragments from these selected elements. Furthermore, we are working on the development of algorithms supporting a fragment insertion method. An additional aspect is the automatic discovery and extraction of choreography fragments based on the identified patterns. Besides the reuse during modeling, there are other use cases where choreography fragments can also be applied. This includes the configurability of choreographies in multi-tenancy scenarios (Hahn et al., 2014) or the refinement of partially specified/defined choreography models (Weiß et al., 2014). In multi-tenancy scenarios, tenants and users register choreography fragments to configure a template choreography model according to their requirements. Furthermore, individual process models not related to a choreography scenario could be connected using a choreography fragment implementing an appropriate service interaction pattern.

ACKNOWLEDGEMENTS

This work is funded by the projects FP7 EU-FET 600792 ALLOW Ensembles and the German DFG within the Cluster of Excellence (EXC310) Simulation Technology.

REFERENCES

Andrikopoulos, V., Bucchiarone, A., Gómez Sáez, S., Karastoyanova, D., and Antares Mezzina, C. (2013). Towards Modeling and Execution of Collective Adaptive Systems. In *Proceedings of WESOA'13*, pages 69–81. Springer.

Andrikopoulos, V., Gómez Sáez, S., Karastoyanova, D., and Weiß, A. (2014). Collaborative, Dynamic & Complex Systems: Modeling, Provision & Execution. In *CLOSER'14*, pages 276–286. SciTePress.

Barros, A., Dumas, M., and ter Hofstede, A. H. (2005). Service interaction patterns. In *Business Process Management*, pages 302–318. Springer.

Decker, G., Kopp, O., and Barros, A. (2008a). An Introduction to Service Choreographies. *Information Technology*, 50(2):122–127.

Decker, G., Kopp, O., Leymann, F., Pfitzner, K., and Weske, M. (2008b). Modeling Service Choreographies using BPMN and BPEL4Chor. In *Proceedings of CAiSE '08*, pages 79–93. Springer.

Decker, G., Kopp, O., Leymann, F., and Weske, M. (2007). BPEL4Chor: Extending BPEL for Modeling Choreographies. In *Proceedings of ICWS '07*, pages 296–303. IEEE.

Decker, G., Kopp, O., Leymann, F., and Weske, M. (2009). Interacting services: from specification to execution. *Data & Knowledge Engineering*, 68(10):946–972.

Eberle, H., Leymann, F., Schleicher, D., Schumm, D., and Unger, T. (2010). Process Fragment Composition Operations. In *Proceedings of APSCC'10*, pages 1–7. IEEE.

Eder, J., Lehmann, M., and Tahamtan, A. (2006). Choreographies as federations of choreographies and orchestrations. In *Advances in Conceptual Modeling-Theory and Practice*, pages 183–192. Springer.

Gottschalk, F., van der Aalst, W. M. P., and Jansen-Vullers, M. H. (2007). Configurable Process Models - A Foundational Approach. In *Reference Modeling*, pages 59–77. Physica-Verlag HD.

Hahn, M., Gómez Sáez, S., Andrikopoulos, V., Karastoyanova, D., and Leymann, F. (2014). SCE^{MT}: A Multi-tenant Service Composition Engine. In *Proceedings of SOCA'14*, pages 89–96. IEEE.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., and Barreto, C. (2005). Web services choreography description language version 1.0.

Leymann, F. and Roller, D. (2000). *Production Workflow - Concepts and Techniques*. PTR Prentice Hall.

Markovic, I. and Pereira, A. C. (2008). Towards a formal framework for reuse in business process modeling. In *BPM Workshops*, pages 484–495. Springer.

Montesi, F. and Yoshida, N. (2013). Compositional choreographies. In *CONCUR'13 – Concurrency Theory*, pages 425–439. Springer.

Object Management Group (2011a). BPMN 2.0 by Example Version 1.0 (non-normative).

Object Management Group (2011b). Business Process Model and Notation (BPMN) Version 2.0.

Schilling, J. (2014). Specification and Development of Choreography Fragments for a Choreography Designer. Diploma thesis, University of Stuttgart.

Schumm, D., Karastoyanova, D., Kopp, O., Leymann, F., Sonntag, M., and Strauch, S. (2011). Process Fragment Libraries for Easier and Faster Development of Process-based Applications. *J. of Sys. Integration*, 2(1):39–55.

Schumm, D., Leymann, F., Ma, Z., Scheibler, T., and Strauch, S. (2010). Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls. In *Proceedings of MKWI'10*, pages 2125–2137. Universitätsverlag Göttingen.

Sonntag, M. and Karastoyanova, D. (2012). Ad hoc Iteration and Re-execution of Activities in Workflows. *Int. J. On Advances in Software*, 5(1 & 2):91–109.

Wagner, S., Kopp, O., and Leymann, F. (2011). Towards Choreography-based Process Distribution In The Cloud. In *Proceedings of CCIS'11*, pages 490–494. IEEE.

Weber, B., Reichert, M., and Rinderle-Ma, S. (2008). Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data Knowl. Eng.*, 66(3):438–466.

Weiß, A., Gómez Sáez, S., Hahn, M., and Karastoyanova, D. (2014). Approach and Refinement Strategies for Flexible Choreography Enactment. In *Proceedings of OTM'14*, pages 93–111. Springer.

Weiß, A. and Karastoyanova, D. (2014). Enabling coupled multi-scale, multi-field experiments through choreographies of data-driven scientific simulations. *Computing*, pages 1–29.

Weske, M. (2012). *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer.