Data-Aware Interaction in Distributed and Collaborative Workflows: Modeling, Semantics, Correctness

David Knuplesch and Rüdiger Pryss and Manfred Reichert
Institute of Databases and Information Systems
Ulm University, Germany
Email: {david.knuplesch, ruediger.pryss, manfred.reichert}@uni-ulm.de

Abstract—IT support for distributed and collaborative workflows as well as related interactions between business partners are becoming increasingly important. For modeling such interactions as flow of message exchanges, different top-down approaches, covered under the term interaction modeling, are provided. Like for workflow models, correctness constitutes a fundamental challenge for interaction models; e.g., to ensure the boundedness and absence of deadlocks and lifelocks. Due to their distributed execution, in addition, interaction models should be message-deterministic and realizable, i.e., the same conversation (i.e., sequence of messages) should always lead to the same result, and it should be ensured that partners always have enough information about the messages they must or may send in a given context. So far, most existing approaches have addressed correctness of interaction models without explicitly considering the data exchanged through messages and used for routing decisions. However, data support is crucial for collaborative workflows and interaction models respectively. This paper enriches interaction models with the data perspective. In particular, it defines the behavior of data-aware interaction models based on Data-Aware Interaction Nets, which use elements of both Interaction Petri Nets and Workflow Nets with Data. Finally, formal correctness criteria for Data-Aware Interaction Nets are derived, guaranteeing the boundedness and absence of deadlocks and lifelocks, and ensuring message-determinism as well as realizability.

I. INTRODUCTION

Workflow management is of utmost importance for companies that want to efficiently handle their workflows as well as their interactions with partners and customers [1]. Despite the varying issues relevant for the IT support of distributed and collaborative workflows [2], common aspects to be considered include the support of appropriate modeling techniques as well as the definition of a formal execution semantics, ensuring proper and correct partner interactions (i.e., message exchanges).

Workflow management methods and techniques tackling these challenges consider a choreography as a specification of message exchanges between the partners of a collaborative workflow. Respective approaches provide a global view on distributed workflows and support partners in correctly defining their private processes (partner processes for short). The latter can be transformed into distributed, executable workflows. When executing these workflows, their interplay is coordinated in terms of a conversation (i.e., a sequence of exchanged messages) that follows the global behavior specified by the choreography.

Currently, there exist two different paradigms for modeling choreographies: interconnection modeling and interaction modeling. Interconnection modeling uses message exchange as link between partner processes or public views on them. In particular, this paradigm does not allow modeling the message exchange separately from the partner processes. Hence, it is considered as a bottom-up approach. Approaches enabling interconnection modeling include BPMN Collaboration Diagram [3], BPEL4Chor [4], and Compositions of Open Nets [5]. By contrast, interaction modeling provides a top-down approach. An Interaction Model specifies the flow of message exchanges without having any knowledge about the partner processes. Moreover, the models of the partner processes are created taking the interaction model into account. Nevertheless, common interaction models use the same patterns as workflow models (e.g., parallel and conditional branchings), but instead of tasks they refer to the messages exchanged. Approaches enabling interaction modeling include iPBMN [6], BPMN Choreography Diagrams [3], Service Interaction Patterns [7], and WSCDL [8].

This paper focuses on the correctness of interaction models. Related issues discussed in the literature include boundedness and absence of deadlocks and lifelocks, as well as the realizability of interaction models [5], [9]–[11]. Realizability postulates that partners always can compute which messages they must or may send in a given execution context. Fig. 1 (1) outlines a simple example of a non-realizable choreography with four partners A, B, C, and D, and two messages \( m_1 \) and \( m_2 \). This interaction model specifies that after sending message \( m_1 \) from A to B, message \( m_2 \) must be sent from C to D. Obviously, only A or B knows when C must send message \( m_2 \), but C does not have this knowledge. Consequently, this interaction model is not realizable. A necessary precondition for realizability is message-deterministic behavior, i.e., the same conversation (i.e., sequence of messages) should always lead to the same result. An example of an interaction model, which is not message-deterministic, is shown in Fig. 1 (2): this interaction model comprises partners A, B, and C, and messages \( m_1, m_2, m_3, \) and \( m_4 \). After sending the first message \( m_1 \), either the upper or the bottom branch shall be chosen.
In any case, the next message \( m_2 \) must be sent from \( B \) to \( C \). Depending on the branch chosen, however, then \( C \) either must send \( m_3 \) to \( B \) or \( m_4 \) to \( A \). From the perspective of \( C \), it cannot be determined, which of the two interpretations shall be applied. By contrast, \( B \) knows the chosen branch (e.g., the upper one). Hence, \( C \) might send \( m_4 \) to \( A \), while \( B \) waits for \( m_3 \), or vice versa.

A property similar to realizability is clear termination. It requires that a partner always can compute, whether he will be sender or receiver of any messages in the sequel. An example of an interaction model, which is not clearly terminating, is shown in Fig. 1 (3). This interaction model comprises partners \( A, B, \) and \( C \), and messages \( m_1, m_2, m_3, \) and \( m_4 \). After sending the first message \( m_1 \) from \( A \) to \( B \), \( B \) can either send message \( m_2 \) to \( A \) or message \( m_4 \) to \( C \). When choosing the first option (i.e. \( B \) sends \( m_2 \) to \( A \)), \( A \) must send \( m_3 \) to \( C \) afterwards. In turn, when choosing the second option (i.e. \( B \) sends \( m_4 \) to \( C \)), the execution is terminated, although \( A \) may still wait for the arrival of message \( m_2 \). Note, that from the perspective of \( A \) nothing has changed since \( m_1 \) was sent.

Existing approaches for interaction modeling do not adequately support the data perspective. Either related execution semantics completely ignore the data perspective or there is a lack of appropriate correctness criteria, especially if routing decisions are based on message data. This paper deals with fundamental correctness issues when making interaction models data-aware. Section II provides an example from the healthcare domain to emphasize the need of data-awareness in interaction models. Section II further discusses the challenges to be tackled when considering the data perspective. Section III then introduces our formal framework for data-aware interaction modeling. First, an interaction meta-model is provided in terms of the Data-Aware Choreography (DAChor). The behavior of a DAChor is described by a transformation to Data-Aware Interaction Nets (DAI Nets). These combine Interaction Petri Nets [9] and Workflow Nets with Data [12]. Based on Data-Aware Interaction Nets, the set of allowed conversations (i.e., message exchanges) is derived and used to introduce formal correctness criteria for DAI Nets and DAChor respectively. These criteria guarantee for the boundedness and absence of deadlocks and lifelocks, and ensure message-determinism, realizability, and clear termination. Section IV discusses related work and Section V concludes with a summary and outlook.

II. Example, Challenges, Contribution

This section introduces a simplified real-world scenario, which we elaborated in the context of case studies conducted in the healthcare domain. These case studies highlighted the relevance of the data perspective in interaction models. Thus, the scenario we select emphasizes the challenges arising from the support of data-awareness in interaction models. It describes the transport of a patient to and from a unit performing a Positron Emission Tomography (PET) scan. A PET scan is a kind of nuclear medicine imaging not performed by the respective hospital itself in our scenario. Thus, if a PET scan is ordered for a patient, patient transportation to the respective provider is required. In this context, the hospital must inform the provider of the PET scan about the patient’s status, such that he can decide on the preparations required. Furthermore, we require a patient to be examined just before the transport to exclude potential risks (e.g., the patient being in a critical condition).

The scenario involves three partners, i.e., the Hospital responsible for the patient and ordering the PET scan, the Transportation (Transp.) Provider transporting the patient, and the PET provider performing the PET scan. The interaction starts with the Hospital requesting the PET scan (Request PET). In the context of this request, the Hospital informs the PET Provider about the status of the patient. In turn, the PET provider confirms the time for which the scan is scheduled (Confirm), and then requests the Transp. Provider to perform the transport (Request Transp.).

- If the patient is in a critical condition, the Transp. Provider requests the Hospital to examine him to check whether he is transportable (Request Exam.). Based on the Result of this examination, the Hospital informs the Transp. Provider on whether to continue or abort the interaction.
- If the interaction is continued or the patient is not in a critical condition, Transp. Provider informs the PET provider after picking up the patient and arriving at the PET unit (Arrival). After the PET scan is performed, the PET provider requests retransport of the Transp. Provider (Retransport). Finally, the Transp. Provider informs the Hospital about the return of the patient (Return).

Obviously, properly modeling the interactions of this scenario requires support for routing decisions based on the data of the messages exchanged. More precisely, in the given scenario, there is a decision referring to data of the first message exchanged (i.e. whether or not the patient is in a critical condition). Another decision refers to the message sent by the hospital and indicating whether the request shall be canceled. Hence, we use a notation based on BPMN 2.0 [3] and iBPMN [9], but enrich it with so-called virtual data objects. We denote this notation as Data-Aware Choreography (DAChor) and use it to model our scenario in Fig. 2. Virtual data objects have a data domain and can be used as variables when defining
conditions for routing decisions. However, these virtual data objects are not used for modeling information flow. Thus, the data assignment relation denotes which data of an interaction is assigned to any virtual data object. Note that such a data assignment relation can only lead from an interaction to a virtual data object, but not vice versa. Furthermore, an interaction is assigned to a message class denoting the message type. From the message class, the sender, receivers, and data domain are inherited (e.g., boolean). Finally, when executing a choreography, messages of the related message class correspond to interactions.

Having a closer look at our scenario, one can recognize that it neither ensures realizability nor clear termination. If the Hospital requests canceling the PET scan, the PET provider is not informed accordingly and hence may still wait for the message; i.e., no clear termination is ensured. However, if Alternative 2 (cf. Fig. 2) is applied, the PET provider will be informed and clear termination can be ensured. In turn, realizability is violated for the given interaction model, since Transp. Provider does not know whether the patient is in a critical condition. Thus, Transp. Provider cannot determine whether an examination must be requested. To ensure realizability, it is not sufficient to only check whether this information was directly sent to Transp. Provider. Consider Alternative 1, which ensures realizability by also sending the confirmation to Transp. Provider, if the patient is in a critical condition. Obviously, implicit knowledge of Transp. Provider about the value of virtual data object Status is sufficient to ensure realizability. This makes the definition of proper correctness criteria for data-aware interaction models Section III very challenging.

Before defining correctness criteria for DAChors, their behavior has to be formalized. In [9], Decker et al. define the behavior of iBPMN choreographies based on their transformation to Interaction Petri Nets (IP Nets). However, IP Nets are unaware of data. This raises the challenge to first enrich IP Nets as well as their behavior with data, i.e., to design Data-Aware Interaction Nets (DAI Nets). Following this, an appropriate transformation is presented.

The main contribution of this paper is to introduce a formal framework for data-aware interaction models in distributed and collaborative settings putting emphasis on correctness. Especially, this framework comprises specific correctness criteria for interaction models (e.g., realizability, clear termination). Note, that the latter exceed traditional correctness and soundness criteria that are known from workflows and interconnection models [5], [13], [14]. Further contributions include the introduction of DAChors and DAI Nets as well as the transformation from DAChors to DAI Nets with well defined behavior.

III. FORMAL FRAMEWORK

This section introduces our formal framework for ensuring correctness of data-aware interaction models. First, the scope of an interaction model is described as interaction domain and in terms of messages (cf. Def. 1 and 2 in Section III-A). Second, Data-Aware Choreographies (DAChors) are introduced as formal meta-model for data-aware interaction modeling (cf. Def. 3 in Section III-B). In Section III-D, the semantics of DAChors is described based on their transformation to Data-Aware Interaction Nets (DAI Nets). DAI Nets combine Interaction Petri Nets (IP Nets) [9] and Workflow Nets with Data (WFD Nets) [12] (cf. Def. 5 in Section III-C). Their behavior is described in terms of markings and execution traces (cf. Def. 8–10 in Section III-E). Def. 12 introduces conversations representing the observable parts of an execution trace (i.e., exchanged messages). Finally, partner views are defined (cf. Def. 14). Based on traces, conversations, and views, we then introduce correctness criteria for DAI Nets and DAChors respectively.
A. Interaction Domains and Messages

This section defines the basic elements of data-aware interaction modeling in terms of an interaction domain. The latter contains roles to differentiate the partners as well as message classes and related data domains. Furthermore, the notion of message (cf. Def. 1 and 2 and Example 1).

Definition 1 (Interaction Domain). An interaction domain is a tuple $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, \text{dom}_C, s_C, r_C, \epsilon)$, with

- $\mathcal{R}$ is a set of roles,
- $\mathcal{D}$ is a set of data domains; each $D \in \mathcal{D}$ represents a finite set of values,
- $C$ is a set of message classes,
- $\text{dom}_C : C \to \mathcal{D}$ is a function assigning to each message class a data domain,
- $s_C : C \to \mathcal{R}$ assigns the sender to each message class,
- $r_C : C \to 2^\mathcal{R}$ assigns the set of receivers to each message class,
- $\epsilon$ is the empty value.

Further, we define $\Omega_\mathcal{I} := \{\epsilon\} \cup \bigcup_{D \in \mathcal{D}} D$ as the set of all values.

Based on Def. 1, Def. 2 introduces the notion of message. A message constitutes an instance of a message class. Furthermore, we introduce several sets of messages.

Definition 2 (Messages). Let $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, \text{dom}_C, s_C, r_C, \epsilon)$ be an interaction domain. Then: A message in $\mathcal{I}$ is a tuple $\mu = (c, x) \in C \times \Omega_\mathcal{I}$, with

- $c \in C$ is the corresponding message class, and
- $x \in \text{dom}_C(c)$ is the message content transferred.

Furthermore, we define:

- $\Sigma_c := \{(c', x) \in C \times \Omega_\mathcal{I} | c' = c \land x \in \text{dom}_C(c')\}$ as set of all messages corresponding to message class $c \in C$,
- $\Sigma_\mathcal{I} := \bigcup_{c \in C} \Sigma_c$ as set of all messages corresponding to interaction domain $\mathcal{I}$,
- $\Sigma_{R^+} := \{(c, v) \in \Sigma_\mathcal{I} | s_C(c) = R\}$ as set of all messages sent by role $R \in \mathcal{R}$,
- $\Sigma_{R^-} := \{(c, v) \in \Sigma_\mathcal{I} | r_C(c) \in R\}$ as set of all messages received by role $R$,
- $\Sigma_R := \Sigma_{R^+} \cup \Sigma_{R^-}$ as set of all messages corresponding to role $R$, i.e. sent or received by $R$.

B. Data-Aware Choreography

Based on the interaction domain from Def. 1, we define the notion of data-aware choreography (DAChor). DAChor enriches BPMN choreography models with virtual data objects, a data assignment relation, and guards.

Definition 3 (Data-Aware Choreography; DAChor). Let $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, \text{dom}_C, s_C, r_C, \epsilon)$ be an interaction domain. Then: A Data-Aware Choreography (DAChor) over $\mathcal{I}$ is a tuple $\mathcal{DAC} = (N, I, G, e_s, E_e, G^s, G^w, G^{\text{dom}V}, G^{\text{dom}G}, V, \text{class}, \to, \to, \text{dom}_V, \text{grd})$, with
Example 1 (Basic Notions). Consider the interaction model of the patient transportation scenario from Fig. 2. Its interaction domain is \( \mathcal{I} = (R, D, C, dom_c, s_c, r_c, \epsilon) \) with:

\[
\begin{align*}
R &= \{ \text{Hospital, PET Provider, Transp. Provider} \} \\
D &= \{ \text{Date} = \{1.1.1900, \ldots, 31.12.2099\} \} \\
C &= \{ \text{Request PET, Confirmation, Request Trans.}, \text{Request Exam.}, \text{Result}, \text{Arrival}, \text{Retransport}, \text{Return}, \text{Confirmation}, \text{Cancel PET} \} \\
s_c(\text{Confirmation}) &= \text{PET provider} \\
s_c(\text{Request PET}) &= \text{Hospital} \\
s_c(\text{Request Trans.}) &= \text{Transp. Provider} \\
s_c(\text{Request Exam.}) &= \text{Hospital} \\
s_c(\text{Result}) &= \text{Hospital} \\
s_c(\text{Arrival}) &= \text{Transp. Provider} \\
s_c(\text{Retransport}) &= \text{Transp. Provider} \\
s_c(\text{Return}) &= \text{Hospital} \\
s_c(\text{Cancel PET}) &= \text{Transp. Provider} \\
\text{dom}_c(\text{Confirmation}) &= \text{Date} \\
\text{dom}_c(\text{Request PET}) &= \text{Status} \\
\text{dom}_c(\text{Request Trans.}) &= \text{Date} \\
\text{dom}_c(\text{Request Exam.}) &= \text{Date} \\
\text{dom}_c(\text{Result}) &= \text{Date} \\
\text{dom}_c(\text{Arrival}) &= \text{Date} \\
\text{dom}_c(\text{Retransport}) &= \text{Date} \\
\text{dom}_c(\text{Cancel PET}) &= \text{Date} \\
\Sigma_e &= \{ \text{(Request PET, uncritical), (Request PET, critical), (Result, abort), (Result, continue), (Result, e), (Arrival, e), (Confirmation, 1.1.1900), \ldots, (Confirmation, 31.12.2099), (Confirmation, 1.1.1900), \ldots, (Confirmation, 31.12.2099), (Result Trans., 1.1.1900), \ldots, (Result Trans., 31.12.2099), (Retransport, e), (Return, e), (Cancel PET, e) \}
\end{align*}
\]

- \( N \) is the set of nodes being the disjoint conjunction of the set of interactions \( I \) and the set of gateways and events \( G \).
- In turn, the latter is the disjoint conjunction of the start event \( \{e_s\} \), the set of end events \( E_e \), the set of AND-splits \( G^a_v \), the set of AND-mergers \( G^m_v \), the set of data-based XOR-splits \( G^s_{dx} \), the set of event-based XOR-splits \( G^s_{ex} \), and the set of AND-mergers \( G^m_v \).
- \( V \) is the set of virtual data objects,
- \( class : I \rightarrow C \) assigns a message class to each interaction,
- each \( \rightarrow \in \mathcal{I} \) is the interaction flow relation,
- each \( \rightarrow \in I \times V \) is the data assignment relation,
- each \( dom_V : V \rightarrow D \) is a function assigning a domain to each virtual data object,
- \( grd : \rightarrow \rightarrow G_v \) is a function assigning a guard to each interaction flow.

The set of guards \( G_v \) is defined in the set as the propositional logic formulas over propositions of the form \( v = s \) or \( v \in \{s_1, s_2, \ldots, s_s\} \). Thereby, \( v \in V \) is a virtual data object and \( s, s_1, s_2, \ldots, s_s \in dom_V(v) \) are values of the related data domain. If a guard \( g \in G_v \) uses a virtual data object \( v \in V \), we denote this as \( v \sim g \). Note that a guard can be constantly true. In this case, we omit it in the graphical representation of the DAChor (cf. Fig. 2). In the following, we introduce the well-formedness of DAChors. Example 2 then provides a formal description of our scenario from Fig. 2.

Definition 4 (Well-Formed DAChor). A DAChor is well-formed, iff the following properties hold:

- the start event, each interaction, and each merge node have exactly one successor, i.e., \( \forall n \in \{e_s\} \cup I \cup G^m_v \cup G^m_x : |\{ n' \in N | n \rightarrow n' \}| = 1 \)
- each split node has at least one successor, i.e., \( \forall g^l_r \in G^s_{dx} \cup G^s_{ex} : |\{ n \in N | g^l_r \rightarrow n \}| \geq 1 \)
- each end event, each interaction, and each split node have exactly one predecessor, i.e., \( \forall n \in E_e \cup I \cup G^m_v \cup G^m_x : |\{ n' \in N | n \rightarrow n' \}| = 1 \)
- each merge node has at least one predecessor, i.e., \( \forall g^l_r \in G^s_{dx} \cup G^s_{ex} : |\{ n \in N | g^l_r \rightarrow n \}| \geq 1 \)
- each event-based XOR-split is solely followed by interactions, i.e., \( \forall g^s_{dx} \in G^s_{dx} : \{ n \in N | g^s_{dx} \rightarrow n \} \subseteq I \)
- the data of an interaction is solely assigned to variables of the same data domain, i.e., \( \forall i \in I, \forall v \in V : i \rightarrow v \Rightarrow dom_c(class(i)) = dom_V(v) \).
- there is no cycle solely consisting of gateways, i.e., \( \forall g_0, g_1, \ldots, g_n \in G : g_0 \rightarrow g_1 \rightarrow \cdots \rightarrow g_n \rightarrow g_0 \).

C. Data-Aware Interaction Net

We introduce the notion of Data-Aware Interaction Net (DAI Net). It combines IP Nets [9] and WFD Nets [12]: Hence, the main elements of a DAI Net are places and transitions. To add data, these elements are enriched with variables and guards on transitions as known from WFD Nets. Furthermore, DAI Nets allow assigning message classes to transitions. Like in IP Nets, respective transitions are denoted as interaction transitions. Finally, all other transitions are called silent transitions.

Definition 5 (Data-Aware Interaction Net; DAI Net). Let \( \mathcal{I} = (R, I, C, dom_c, s_c, r_c, \epsilon) \) be an interaction domain. Then: A tuple \( \# = (P, p_{in}, P_o, P_f, T, T_S, T_I, V, class, \rightarrow, dom_V, grd) \) is called Data-Aware Interaction Net (DAI Net) over \( \mathcal{I} \), where:

- \( P \) is the set of places; \( P \) can be partitioned into the initial place \( p_{in} \), the set of ordinary places \( P_o \), and the set of final places \( P_f \).
- \( T \) is the set of transitions; \( T \) can be partitioned into the sets of silent transitions \( T_S \) and the set of interaction transitions \( T_I \).
Example 2 (DAChor). Consider the scenario from Fig. 2. Basing its interaction domain \( \mathcal{I} \) (cf. Example 1) we can describe the given scenario as follows: \( \text{DAChor} \) \( \mathcal{D} = (N, I, G, e, s, G_e^i, G_s^i, G_e^m, G_s^m, V, \text{class}, \rightarrow, \rightarrow, \text{dom}_V, \text{grd}) \):

\[
\begin{aligned}
&I = \{i_1, \ldots, i_6\}, E_6 = \{e_{1i}, e_{2i}\}, G_e = \{g_{e1}, g_{e2}\}, G_s = \{g_{s1}, g_{s2}\},
&V = \{s_1, s_2\}, V = \{\text{status}, \text{order}\}.
\end{aligned}
\]

\[
\begin{aligned}
&\rightarrow = \{(e_{1i}, i_1), (e_{1i}, i_2), (e_{1i}, i_3), (e_{1i}, i_4), (e_{1i}, i_5), (e_{1i}, i_6),
&(e_{2i}, i_1), (e_{2i}, i_2), (e_{2i}, i_3), (e_{2i}, i_4), (e_{2i}, i_5), (e_{2i}, i_6)\}.
\end{aligned}
\]

Further, we define \( \text{class}(t_i) = \text{Request PET} \) and \( \text{class}(t_i) = \text{Arrival} \) for each interaction transition, \( e \rightarrow \Rightarrow \) is the flow relation, \( \rightarrow \subseteq \) is the data assignment relation. It expresses that the data of an interaction transition is assigned to the related variable, \( \text{dom}_V : V \rightarrow \mathcal{D} \) is a function assigning a data domain to each variable, \( \text{grd} : T \rightarrow \mathcal{G} \) is a function assigning a guard (cf. Def 3) to each interaction flow relation.

Further, we define:

- \( V \) is the set of variables,
- \( \text{class} : T_i \rightarrow C \) is a function assigning a message class to each interaction transition,
- \( \rightarrow \subseteq (\mathcal{P}(P_f) \times T) \cup (T \times (P - \{p_{in}\})) \) is the flow relation,
- \( \rightarrow \subseteq T_i \times V \) is the data assignment relation. It expresses that the data of an interaction transition is assigned to the related variable,
- \( \text{dom}_V : V \rightarrow \mathcal{D} \) is a function assigning a data domain to each variable,
- \( \text{grd} : T \rightarrow \mathcal{G} \) is a function assigning a guard (cf. Def 3) to each interaction flow relation.

Definition 6 (Well-Formed DAI Net). A DAI Net is well-formed, if the following properties hold:

- each transition has at least one incoming and one succeeding place, i.e., \( \forall t \in T : \exists p_1, p_2 \in P : p_1 \rightarrow t \rightarrow p_2 \)
- the content of an interaction transition is solely assigned to variables of the same data domain, i.e., \( \forall t_i \in T_i, \forall v \in V : t_i \rightarrow v \Rightarrow \text{dom}_C(\text{class}(t_i)) = \text{dom}_V(v) \)
- there exists no cycle solely consisting of places and silent transitions, i.e., \( \exists p_0, p_1, \ldots, p_n \in P : p_0 \rightarrow t_0 \rightarrow p_1 \rightarrow t_1 \rightarrow \cdots \rightarrow p_n \rightarrow t_n \rightarrow p_0 \)

D. Mapping DACHors to DAI Nets

In Section III-C, we introduced DAI Nets to define the behavior of DACHors. Based on this we can now define a mapping from data-aware choreographies to DAI Nets. This mapping is based on the approach proposed by Decker et al. [9] who define the behavior of iBPMN Choreographies through their transformation to IP Nets.

Definition 7 (Mapping DACHors to DAI Nets). Let \( \text{DAChor} = (N, I, G, e, s, G_e^i, G_s^i, G_e^m, G_s^m, V, \text{class}, \rightarrow, \rightarrow, \text{dom}_V, \text{grd}) \) be a DACHor (cf. Def. 3). Then, \( \text{DACHor} \) can be mapped to a DAI Net defined as \( \# = (P, p_{in}, P_f, T, T_S, T_I, V, \text{class}, \rightarrow, \rightarrow, \text{dom}_V, \text{grd}) \) with

- \( P = \{p_1, p_2\} \) if \( \text{dom}_V(v) \)
- \( \text{dom}_V(v) \)

Theorem 1 states that the mapping from DACHors to DAI Nets preserves well-formedness as proven in [15]. The application to our example is shown in Example 3 and Fig. 4.

Theorem 1 (Preservation of Well-Formedness). Let \( \text{DAChor} \) be a DACHor that is mapped to a DAI Net \( \# \). If \( \text{DAChor} \) is well-formed, \( \# \) is well-formed as well.

E. Behavior of DAI Nets

Since DAI Nets are based on both WFD Nets and IP Nets, we use token semantics (i.e., tokens assigned to places and token changes) to define their behavior. Together with the values of variables, tokens define the marking of a DAI Net. Each Interaction Net starts with an initial marking, with exactly one token placed in the initial place \( p_{in} \) and each variable having the empty value \( \epsilon \). A marking is called final, if all tokens belong to final places of \( P_f \). A transition \( t \) is activated under marking \( m \), iff all directly preceding places of \( t \) contain at least one token, and the guard of \( t \) is evaluatable and evaluates to true.

Definition 8 (DAI Net Markings and Activated Transitions). Let \( \# = (P, p_{in}, P_f, T, T_S, T_I, V, \text{class}, \rightarrow, \rightarrow, \text{dom}_V, \text{grd}) \) be a DAI Net. Then: A marking of \( \# \) is a tuple \( m = (\emptyset, \text{val}) \) with

- \( \emptyset : P \rightarrow \mathbb{N}_0 \) assigns to each place the number of corresponding tokens,
- \( \text{val} : V \rightarrow \mathcal{O}_2 \) assigns to each variable its current value,
- \( \text{val}(v) \) is either the empty value \( \epsilon \) or an element of the variable’s domain, i.e., \( \text{val}(v) \in \text{dom}_V(v) \cup \{\epsilon\} \).

Additionally, for each DAI Net \# we define the

- set of all markings \( M_{\#} \), whereby
- \( M_{\#} = \{m = (\emptyset, \text{val}) \mid m \text{ is a marking of } \#\} \)
Example 3 (Transformation). The DACChor $DAC = (N, I, G, E, E_c, G^0, G^1, G^2, G^3, G^4, V, class, \rightarrow, dom, V, \text{grd})$ from our Example 2 (i.e., the patient transportation scenario) is mapped to the DAI Net $\# = (P, P_{in}, P_{o}, P_{f1}, T, T_{S}, T_{I}, V, \text{class'}, \rightarrow', \text{dom', \text{grd'}})$ as follows (cf. Fig. 4):

\[
P_{in} = \{P_{in,x}, \ldots, P_{in,3}\}, \quad P_{o} = \{P_{o,1}, \ldots, P_{o,3}\}, \quad P_{f1} = \{P_{f1,1}, \ldots, P_{f1,3}\}
\]

\[
T_S = \{(t_{in,x+1}, t_{in,x+1}), (t_{in,x+1}, t_{in,x+1}), (t_{in,x+1}, t_{in,x+1}), \}
\]

\[
V = \{\text{Status, Order}, \rightarrow = t_{1x}, t_{2x}, \ldots, t_{nx}\}
\]

Further, $\# \mathrel{\subseteq} M_\# \times T$ is the transition activation relation. $m \mathrel{\rightarrow} t$ denotes that marking $m \in M_\#$ activates transition $t \in T$, iff the following conditions hold:

1. $\forall p \in P_{\rightarrow} : (\emptyset)(p) \geq 1$.
2. $\forall v \in V : \text{val}(v) \neq \epsilon$.
3. $\text{grd}(t)$ is satisfied for marking $m$.

If a transition is activated, it may fire and lead from the current marking to a subsequent one. More precisely, one token is taken from each preceding place and one is added to each succeeding place. Silent transitions fire immediately when they become activated. Activated interaction transitions fire, if and only if a message of the corresponding message class is sent. In this case, the value of the message is assigned to virtual data objects as expressed by the data assignment relation. Note that a message can only be sent if an interaction transition of the related message class is activated and no silent transition is activated (cf. Def. 9).

Definition 9 (Options and Subsequent Markings of DAI Nets). Let $\# = (P, P_{in}, P_{o}, P_{f1}, T, T_{S}, T_{I}, V, \text{class,} \rightarrow, \text{dom, \text{grd'}})$ be a DAI Net, an $m = (\emptyset, \text{val}) \in M_\#$ be two related markings. Then:

1. $\emptyset \mathrel{\subseteq} T_S \mathrel{\cup} \Sigma_\#$ is the set of all options on $\#$.
2. $\text{opt}_\# : M_\# \rightarrow 2^O_\# : m \mapsto \{o \in O_\# \mid m \mathrel{\rightarrow} m'\}$ maps each marking $m$ to the options available under $m$.
3. $m \mathrel{\rightarrow} m'$ expresses that $m$ leads to $m'$ by applying option $o \in \text{opt}_\#(m)$ with:
   - Case 1: $o = t_{sx} \in T_S$ is a silent transition. Then: $m \mathrel{\rightarrow} t_s m'$ holds, iff each of the following conditions is met:
     1. $m \mathrel{\rightarrow} t_s m'$
     2. $\forall p \in P_{\rightarrow} : (\emptyset)(p) = (\emptyset)(p) - 1$.

Fig. 4. DAI Net derived for the patient transportation scenario
Based on Def. 9, the following two theorems can be derived. (See [15] for formal proofs).

**Theorem 2** (Separation of Options). Let \( \# \) be a DAI Net. Then: For each marking, the set of options either contains solely silent transitions or messages or it is empty, i.e., \( \forall m \in M_\# : \text{opt}_\#(m) \neq \emptyset \Rightarrow \text{opt}_\#(m) \subseteq T_S \cup \text{opt}_\#(m) \subseteq \Sigma_\# \)

**Theorem 3** (Termination of final markings). Let \( \# \) be a DAI Net. Then: For each final marking, the set of options is empty, i.e., \( \forall m \in F_\# : \text{opt}_\#(m) = \emptyset \).

Based on Def. 9, we define traces on DAI Nets as sequences of options. To be more precise, a trace corresponds to a related sequence of markings that starts with the initial marking. If this related sequence of markings ends with a final marking, we denote the trace as completed.

**Definition 10** (Traces, Prefixes, and Extensions). (A) Let \( \# \) = \((P, p_{in}, P_0, P_f, T, T_S, T_I, V, class, \rightarrow, \rightarrow, dom_v, \text{grd})\) be a DAI Net and \( \tau = (t_k)_{k \in [1..n]} \in O_\#^* \) be a finite sequence of options (i.e. silent transitions and messages) with length \( |\tau| = n \in \mathbb{N} \). Let further \( m = (m_k)_{k \in [1..n+1]} \in M_\#^* \) be a finite sequence of markings with length \( n + 1 \). Then:

- \( \tau \triangleright m \) denotes that \( \tau \) and \( m \) are related sequences, iff \( \forall t_i \in [1..n] : m_i \overset{t_i}{\rightarrow} m_{i+1} \) and \( m_1 = m_\#^\triangleright \).
- \( \text{last} : M_\#^* \rightarrow M_\# \) with \( (m_k)_{k \in [1..n]} \triangleright m_\#^\triangleright m_\# \) is a function mapping a sequence of markings to its last marking.
- \( \tau \triangleleft \) is a trace, iff \( \exists m \in M_\#^* \) and \( \tau \triangleright m \). If \( \text{last}(m) \in F_\# \), we denote \( \tau \) as completed trace.
- \( T_\# \) denotes the set of all traces on \( \# \).
- \( T_\#^\triangleright \) denotes the set of all completed traces on \( \# \).

(B) Let \( L \subseteq M^* \) be a set of finite sequences over a set \( M \) and let \( \alpha = (a_k)_{k \in [1..n]}, b = (b_k)_{k \in [1..l]}, c = (c_m)_{m \in [1..m]} \in L \) be elements of \( L \), i.e. sequences over \( M \). Then:

- \( a \preceq b \) (\( a \preceq b \) denotes \( a \) is prefix (real prefix) of \( b \) and \( b \) an extension (real extension) of \( a \) iff \( n \leq l \) and \( \forall i \in [1..l] : a_i = b_i \).
- \( a \overset{c}{\rightarrow} b \) denotes that \( a \) is extended by \( c \) to \( b \), iff \( m + n = l \) and \( \alpha = b \overset{c}{\rightarrow} b \).
- \( L^\triangleright b := \{ a \in L | a \preceq b \} \) (\( L^\triangleright b := \{ a \in L | a \preceq b \} \) denotes the subset of \( L \) that contains all prefixes (real prefixes) of \( b \) in \( L \), and \( L^\triangleright b := \{ a \in L | a \preceq b \} \) denotes the subset of \( L \) that contains all extensions (real extensions) of \( b \) in \( L \).)

We described the behavior of a DAI Net by means of its traces. We can also use traces to characterize the desired behavioral properties of DAI Nets. The first one is **determinism**. It expresses that a trace is unique in terms of its related markings, i.e., replaying a trace will always lead to the same marking.

The second fundamental property is **soundness** in terms of boundedness as well as the absence of deadlocks and lifecycles [16].

**Definition 11** (Determinism and Soundness). (A) We call a DAI Net \( \# \) **deterministic**, iff for each trace \( \tau \) on \( \# \) there exists exactly one related sequence of markings, i.e., \( \forall \tau \in T_\# : |\{ m \in M_\#^* | m \sim_\# \tau \}| = 1 \).

Let \( \# \) be a deterministic DAI Net. Then: Function \( \text{mark}_\# \) maps each trace on \( \# \) to its current marking, i.e., the last marking of the related sequence of markings:

\[ \text{mark}_\# : T_\# \rightarrow M_\# : \tau \mapsto \text{mark}_\#(\tau) := \text{last}(m) \],

whereby \( m \) is defined by \( \tau \sim_\# : m \in M_\#^* \); Since \( \# \) is deterministic, the definition of \( m \) is unique. Thus, \( \text{mark}_\# \) is well defined.

(B) We call a deterministic DAI Net \( \# \) **sound**, iff the following conditions hold:

- There exist completed traces on \( \# \), i.e., \( T_\#^\triangleright \neq \emptyset \).
- Each trace on \( \# \) is a prefix of a completed trace, i.e., \( \forall \tau \in T_\# : \exists \tau' \in T_\#^\triangleright : \tau \subseteq_\# \tau' \).
- The set of reachable markings is finite, i.e., \( \left| \{ m \in M_\# | \exists \tau \in T_\#: \text{last}(\tau) = m \} \right| \in \mathbb{N} \).

Note that the observable behavior of any DAI Net is solely explained through the messages exchanged. Hence, we must abstract from the silent elements of traces (i.e. silent transitions) and define the observable behavior as a conversation being the projection of a trace to its messages (i.e., the part of the trace defining its semantic). In the following, we first introduce projections of sequences.

**Definition 12** (Projections and Conversations). Let \( A,B \) be two sets with \( B \subseteq A \), and \( \# = (P, p_{in}, P_0, P_f, T, T_S, T_I, V, class, \rightarrow, \rightarrow, dom_v, \text{grd}) \) be a DAI Net and \( \tau \in T_\# \) be a trace on \( \# \). Then:

- \( \Pi_B : A^* \rightarrow B^* : a \mapsto \Pi_B(a) \) is the projection function that restricts a sequence \( a \in A^* \) to its elements of \( B \).
- \( \eta \in \Sigma_\#^* \) denotes a conversation on \( \# \), if \( \eta \) is the projection of a trace on \( \# \) to its messages, i.e., \( \exists (\tau) \in T_\# : \Pi_{\Sigma_\#}(\tau) = \eta \). \( \eta \) denotes a completed conversation on \( \# \), iff it is the projection of a completed trace on \( \# \).
- \( C_\# \) denotes the set of all conversations on \( \# \).
- \( C_\#^\triangleright \) denotes the set of all completed conversations on \( \# \).
- \( \text{con}_\# : T_\# \rightarrow C_\# : \tau \mapsto \text{con}_\#(\tau) := \Pi_{\Sigma_\#}(\tau) \) maps each trace to the related conversation.

As aforementioned, the behavior of silent transitions is not observable. Hence, to ensure compatible behavior of participating roles, silent transitions must behave deterministically. In other words, it must be possible to determine the behavior of a DAI Net solely based on the messages exchanged, i.e.,
Example 4 (Traces and Conversations). Consider the DAI Net \# from Example 3. Its set of completed traces $T^C_{\#}$ consists of traces $\tau_1$, $\tau_2$, and $\tau_3$. Projecting them to their messages leads to the conversations $\eta_1$, $\eta_2$, and $\eta_3$, which build $C_{\#}$:

$\tau_1 = \langle \text{Request FET}, \text{uncritical}, \text{Confirmation}_1 \rangle, \langle \text{Request Trans}_1 \rangle, t_2^1, t_3^1, t_4^1, t_5^1, (\text{Arrival}, e), (\text{Retransport}, e), (\text{Return}, e) >$

$\tau_2 = \langle \text{Request FET}, \text{critical}, \text{Confirmation}_2 \rangle, \langle \text{Request Trans}_2 \rangle, t_2^2, t_3^2, t_4^2, t_5^2, (\text{Request Exam}, e), (\text{Result}, \text{abort}), (\text{Retransport}, e), (\text{Return}, e) >$

$\tau_3 = \langle \text{Request FET}, \text{critical}, \text{Confirmation}_3 \rangle, \langle \text{Request Trans}_3 \rangle, t_2^3, t_3^3, t_4^3, t_5^3, (\text{Request Exam}, e), (\text{Result}, \text{continue}), t_2^1, t_3^1, t_4^1, t_5^1, (\text{Arrival}, e), (\text{Retransport}, e), (\text{Return}, e) >$

message-determinism. First, this requires, that firing of silent transitions always terminates, i.e., it is impossible to solely execute silent transitions infinitely (cf. Theorem 4 and see [15] for formal proof). Second, when silent transitions terminate, the set of activated options may only depend on the messages exchanged before, i.e., it should be independent from the order in which the silent transitions were fired.

Theorem 4 (Termination of silent subtraces). On a well-formed DAI Net \#, any trace cannot infinitely be continued by silent transitions, i.e.,

$\forall \tau \in T^C_{\#} : \exists N \in \mathbb{N} \text{ such that } \forall \nu \in T^C_{\#} \text{ with } |\tau| + N < |\nu| \Rightarrow \text{con}_{\#}(\tau) \neq \text{con}_{\#}(\nu).$

According to Theorem 4, a DAI Net is message-deterministic, if the set of activated messages solely depends on the messages exchanged before (cf. Def. 13).

Definition 13 (Message-Determinism). We call a deterministic and sound DAI Net \# message-deterministic, iff the same sequence of messages always activates the same messages, i.e., the set of activated messages solely depends on the messages exchanged before, i.e.,

$\forall \tau, \nu \in T^C_{\#} :$

$(\text{opt}_{\#}(\text{mark}_{\#}(\tau)), \text{opt}_{\#}(\text{mark}_{\#}(\nu)) \in \Sigma_{\#} \wedge \Pi_{\#}(\tau) = \Pi_{\#}(\nu)) \Rightarrow \text{opt}_{\#}(\text{mark}_{\#}(\tau)) = \text{opt}_{\#}(\text{mark}_{\#}(\nu))$

Let \# be a deterministic, sound and message-deterministic DAI Net. Then: Function $\text{mark}_{\#}$ maps each conversation to the set of messages it activates:

$\text{mo}_{\#} : C_{\#} \rightarrow 2^\Sigma_{\#} : \eta \mapsto \text{mo}_{\#}(\eta) := \text{opt}_{\#}(\text{mark}_{\#}(\tau)), \tau \in C_{\#} \text{ is defined by } \eta = \text{con}_{\#}(\tau) \text{ and } \text{opt}_{\#}(\text{mark}_{\#}(\tau)) \in \Sigma_{\#}.$

Since \# is message-deterministic, the definition is unique. Thus, $\text{mo}_{\#}$ is well defined.

Until now, we solely considered DAI Nets and conversations from a global perspective. However, a role solely knows those messages of a conversation it sends or receives. Thus, in Def. 14 the view of a role on the messages of a conversation is introduced. Further, for each role the set of activated options is defined.

Definition 14 (Views on Conversations and Options).

Let $\mathcal{I} = (R, D, C, \text{dom}_C, \text{sc}_C, \text{rc}_C, e)$ be an interaction domain and let the tuple \# = (P, P_{in}, P_o, P_{fi}, T, T_S, T_I, V, \text{class}, \rightarrow, \text{dom}_V, \text{grd}) be a sound, deterministic, and message-deterministic DAI Net. Let further $R \in R$ be a role. Then we can define the following views

- $\text{vc}_{\#} : C_{\#} \rightarrow \Sigma^\ast_R : (\eta_k)_{k \in [1..n]} \mapsto \text{vc}_{\#}^R(\eta) := \Pi_{\Sigma_R}(\eta)$ maps each conversation on \# to the view to the messages sent or received by $R$.
- $\text{vc}_{\#}^R : C_{\#} \rightarrow \Sigma^\ast_R : (\eta_k)_{k \in [1..n]} \mapsto \text{vc}_{\#}^R(\eta) := \Pi_{\Sigma_R}(\eta)$ maps each conversation on \# to the messages sent by $R$.
- $\text{vo}_{\#} : 2^\Sigma_{\#} \rightarrow 2^\Sigma_{\#} : M \mapsto \text{vo}_{\#}^R(M) := M \cap \Sigma_R$ maps each set of messages to its messages sent or received by $R$.
- $\text{vo}_{\#}^R : 2^\Sigma_{\#} \rightarrow 2^\Sigma_{\#} : M \mapsto \text{vo}_{\#}^R(M) := M \cap \Sigma_{R-}$ maps each set of messages to its messages sent by $R$ and $\text{dom}_V.$

Based on Def. 14, we can define realizability. It denotes DAI Nets to be deterministic from the viewpoint of a role. Further, clear termination is defined, which indicates that a role can determine when it sent or received its last message.

Definition 15 (Realizability, Clear Termination). Let \# be a deterministic, sound, and message-deterministic DAI Net. Then, for a role $R \in R$:

- \# is realizable, iff the messages role $R$ may send solely depend on the messages $R$ has sent and received before, i.e.,

$\forall R \in R : \forall \eta, \kappa \in C_{\#} : \text{vo}_{\#}^R(\eta) = \text{vo}_{\#}^R(\kappa) \Rightarrow \text{vo}_{\#}^R(\text{mo}_{\#}(\eta)) = \text{vo}_{\#}^R(\text{mo}_{\#}(\kappa))$

- \# clearly terminates, iff it solely depends on the messages $R$ has sent and received before whether further interaction with $R$ will occur, i.e.,

$\forall R \in R : \forall \eta \in C_{\#} \Rightarrow \exists \kappa \in C_{\#} : \text{vo}_{\#}^R(\eta) \not\subset \text{vo}_{\#}^R(\kappa)$

An important issue concerns decidability of the introduced properties of DAI Nets and DAChors; i.e., determinism, soundness, message-determinism, realizability, and clear termination (cf. Def. 11-15). Basically, these properties are decidable. Due to lack of space, we omit a discussion in this paper.

IV. RELATED WORK

In the context of workflows, correctness has been discussed for a long time [16]. The approaches presented [12], [17] consider data as well. The two paradigms for modeling choreographies (i.e. interconnection and interaction models) are compared in [18]. Examples of interconnection models are BPMN 2.0 Collaborations [3] and BPEL4Chor [4]. There are several approaches that discuss the verification classic
soundness criteria (i.e., boundedness, absence of deadlocks, absence and lifelocks) of distributed and collaborative workflows and service orchestrations [5], [13], [14], [19]–[22]. Some approaches use data dependencies to interconnect processes and to define process interactions [23], [24]. Examples of interaction models (i.e., the paradigm we apply) include Service Interaction Patterns [7], WSCDL [8], iBPMN Choreographies [9], and BPMN 2.0 Choreographies [3]. Our approach has been mainly inspired by [9], which defines the behavior of iBPMN Choreographies through their transformation to Interaction Petri Nets and further discusses correctness and realizability. Realizability of interaction models is also discussed in [10], [25]. Furthermore, [11] provides a tool for checking realizability of BPMN 2.0 Choreographies. However, all these approaches do not explicitly consider the data exchanged by messages and used for routing decisions.

In [26], [27], state-based conversation protocols are introduced, which are aware of message contents. The messages (and data) exchanged trigger state transitions. Thus, different data may trigger different transitions. However, conversation protocols do not support the modeling of parallelism since they are state-based. Furthermore, realizability of conversation protocols requires that at every state each partner is either able to send or receive a message or to terminate (autonomy condition). This condition strongly restricts parallelism. For example, consider a choreography solely consisting of two parallel branches: In the upper branch partner A sends a message $m_1$ to partner $B$ and partner $B$ sends message $m_2$ to $A$ in the lower branch. Obviously, the autonomy condition is violated although the choreography is realizable (cf. Def. 15).

Hence, conversation protocols do not constitute interaction models in our point of view. Thus, to our best knowledge the framework presented within this paper is the first one that considers realizability and clear termination of data-aware interaction models.

V. SUMMARY AND OUTLOOK

Our vision is to provide sophisticated support for distributed and collaborative workflows. To foster this vision, we base our work on the analysis of scenarios from different domains. In essence, we learned that data support is practically relevant for interaction models from a variety of domains.

Further, this paper introduced a formal framework for data-aware interaction models and described how correctness can be ensured. The main parts of our framework include DACHors and DAI Nets as well as the transformation of DACHors to DAI Nets. Further, the behavior of DAI Nets is defined. Other fundamental contributions are the definitions of correctness criteria for data-aware interaction models. The latter include message-determinism, realizability, and clear termination. In future work, we will extend our framework to support asynchronous message exchange and related correctness properties. Finally, we will develop algorithms for efficiently checking correctness of data-aware interaction models. In this context, we plan to apply abstraction strategies to large data domains similar to [28].

ACKNOWLEDGMENT

This work was done within the research project C²Pro funded by the German Research Foundation (DFG), Project number: RE 1402/2-1.

REFERENCES