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 and related interactions between business partners is becoming increasingly important. For modeling such partner 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 as well; 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 therefore 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* [1] and *Workflow Nets with Data* [2]. 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 [3]. Despite the varying issues relevant for the IT support of distributed and collaborative workflows [4], common aspects to be considered include the support of appropriate modeling techniques as well as the definition of 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 defining their intra-organizational workflows (*partner processes* for short). The latter can be transformed into distributed, executable workflows. When executing the latter, their interplay is expressed 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 [5], BPEL4Chor [6], and Compositions of Open Nets [7]. 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 iBPMN [8], BPMN Choreography Diagrams [5], Service Interaction Patterns [9], and WSCDL [10].

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 [1], [7], [11], [12]. 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 has to be chosen.

In any case, the next message m_2 must be sent from B to C. Depending on the branch chosen. However, C then 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 may know 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 or not he will be sender or receiver of messages anymore. 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.

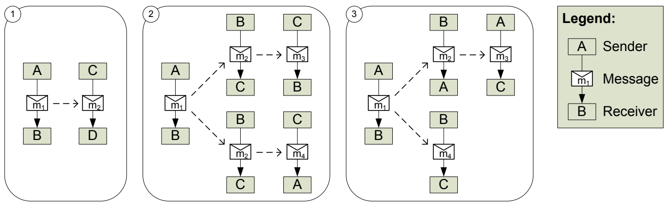

Fig. 1. Violating realizability, message-determinism, clear termination [1]

However, existing approaches for interaction modeling do not adequately support the data perspective. Either related execution semantics 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 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* [1] and *Workflow Nets with Data* [2]. 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. The scenario was elaborated in the context of several case studies we conducted in the healthcare domain. The case studies highlighted the relevance of the data perspective in interaction models. Thus, the scenario we selected 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 provider of the PET scan is required. In this context, the hospital has to inform the provider of the PET scan about the patient's status, such that this provider can decide on preparations required. Furthermore, we require a patient to be examined just before the transport to exclude potential risks, if he is in a critical condition.

This simple scenario involves three partners: 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 when the scan is scheduled (`Confirm`), and then requests the `Transp. Provider` to perform the transport (`Request Trans.`).

- If the patient is in a critical condition, the `Transp. Provider` requests the `Hospital` to examine the patient to check whether he is transportable (`Request Exam.`). Based on the `Result` of this examination, the `Hospital` informs the `Transp. Provider` about, whether to continue or abort the interaction.
- If the interaction is continued or the patient is not in a critical condition, the `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 or not the request shall be cancelled. Hence, we use a notation based on BPMN 2.0 [5] and iBPMN [1], 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

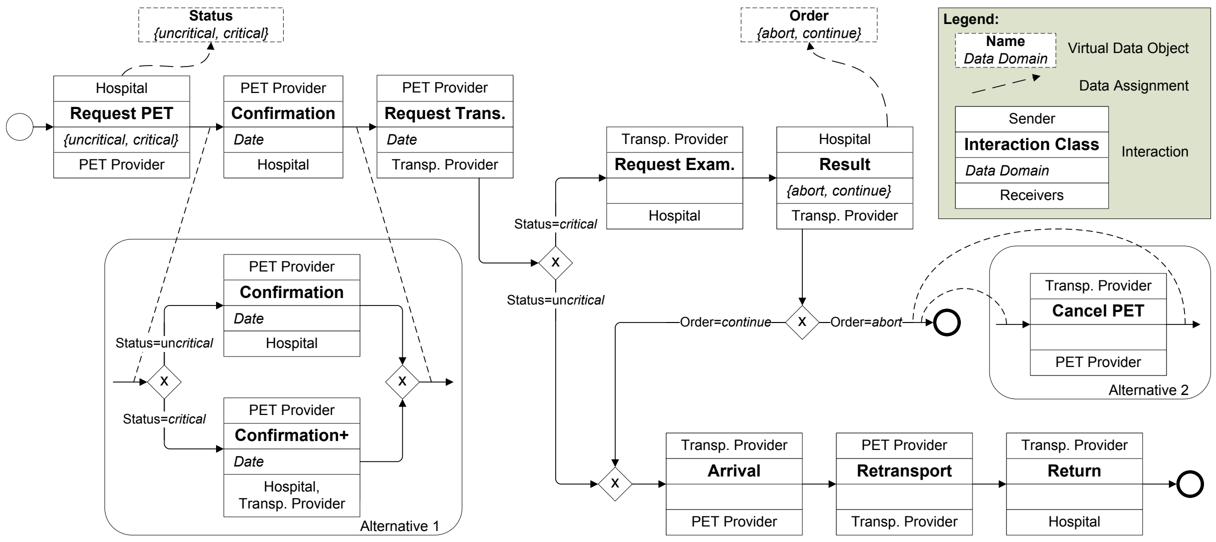

Fig. 2. Patient transportation scenario as DAChor

when defining conditions for routing decisions. Note that these virtual data objects are not used to model information flow. Thus, the *data assignment relation* denotes which data of an interaction is assigned to a 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. Further, an *interaction* is assigned to a *message class* denoting the message type. From the message class, the sender, the receivers, and the 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* is applied, the `PET provider` will be informed and clear termination can be ensured. Further, 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 to request an examination. In order to ensure realizability, it is not sufficient to only check whether this information was directly sent to the `Transp. Provider`. Consider *Alternative 1*, which ensures realizability by also sending the confirmation to `Transp. Provider`, in case the patient is in a critical condition. Obviously, implicit knowledge of `Transp. Provider` about the value of virtual data object `Status` is enough 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 [1], 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 to Data-Aware Interaction Nets (DAI Nets). Following this, an appropriate transformation is introduced.

The main contribution of this paper is to introduce a formal framework for data-aware interaction models in distributed and collaborative settings that puts emphasis on correctness. Especially, our framework considers particular correctness criteria for interaction models (e.g. realizability, clear termination). It should be clear, that the latter exceed classic correctness and soundness criteria that are known from workflows and common service orchestrations through interconnection models [7], [13], [14]. Further contributions include the introduction of DAChors and DAI Nets as well as the transformation from DAChors to DAI Nets and the definition of the behavior of the latter.

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 Def. 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) [1] and *Workflow Nets with Data (WFD Nets)* [2] (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

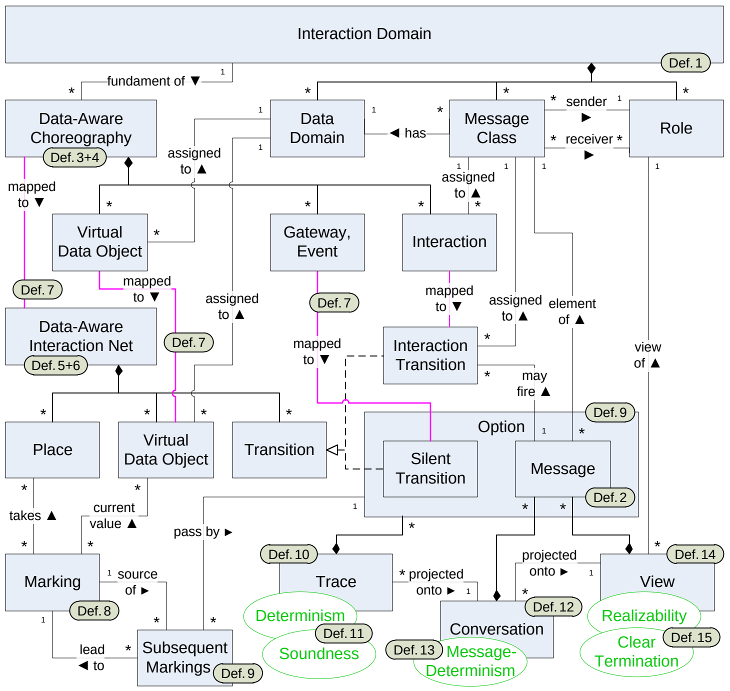

Fig. 3. Overview of our formal framework

(cf. Def. 11, Def. 13, and Def. 15). Fig. 3 provides an overview of the main elements of our formal framework and their relations.

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*. Further, the notion of *message* is introduced as an instance of a message class (cf. Def. 1 and Example 1).

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

- *$\mathcal{R}$ is a set of roles,*
- *$\mathcal{D}$ is a set of data domains, whereby each $D \in \mathcal{D}$ has finitely many values in it*
- *C is a set of message classes,*
- *$dom_C : C \to \mathcal{D}$ 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 mes. cl.,*
- *ϵ 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, we introduce the notion of *message* in Def. 2. A message constitutes an instance of a message class. Furthermore, we introduce several sets of messages.

Definition 2 (Messages). *Let $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, dom_C, s_C, r_C, \epsilon)$ 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*
- *$x \in dom_C(c)$ is the message content transferred.*

Furthermore, we define:

- *$\Sigma_c := \{(c', x) \in C \times \Omega_{\mathcal{I}} \mid c' = c \wedge x \in dom_C(c')\}$ 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 $\mathcal{I}$,*
- *$\Sigma_{R\to} := \{(c, v) \in \Sigma_{\mathcal{I}} | s_C(c) = R\}$ as set of all messages sent by role $R \in \mathcal{R}$,*
- *$\Sigma_{\to R} := \{(c, v) \in \Sigma_{\mathcal{I}} | R \in r_C(c)\}$ as set of all messages received by role R,*
- *$\Sigma_R := \Sigma_{R\to} \cup \Sigma_{\to R}$ 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 $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, dom_C, s_C, r_C, \epsilon)$ be an interaction domain. Then: A Data-Aware Choreography (DAChor) over $\mathcal{I}$ is a tuple $DAC = (N, I, G, e_s, E_e, G^s_+, G^m_+, G^s_{d\times}, G^s_{e\times}, G^m_\times, V, class, \to, \dashrightarrow, dom_V, grd)$, with*

- *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^s_+, the set of AND-mergers G^m_+, the set of data-based*

Example 1 (Basic Notations)**.** *Consider the interaction model from of the patient transportation scenario from Fig. 2. Its interaction domain is $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, dom_C, s_C, r_C, \epsilon)$ with:*

$$
\begin{array}{lcl}
\mathcal{R} & = & \{\texttt{Hospital}, \texttt{PET Provider}, \texttt{Transp. Provider}\} \\
\mathcal{D} & = & \{D_\epsilon = \{\epsilon\}, D_{Status} = \{uncritical, critical\}, D_{Order} = \{abort, continue\}, D_{Date} = \{1.1.1900, \dots, 31.12.2099\} \\
\mathcal{C} & = & \{\texttt{Request PET}, \texttt{Confirmation}, \texttt{Request Trans.}, \texttt{Request Exam.}, \texttt{Result}, \texttt{Arrival}, \texttt{Retransport}, \texttt{Return}, \texttt{Confirmation+}, \texttt{Cancel PET}\}
\end{array}
$$

$$
\begin{array}{lcl|lcl}
s_C(\texttt{Request PET}) & = & \texttt{Hospital} & r_C(\texttt{Request PET}) & = & \{\texttt{PET provider}\} \\
s_C(\texttt{Confirmation}) & = & \texttt{PET provider} & r_C(\texttt{Confirmation}) & = & \{\texttt{Hospital}\} \\
s_C(\texttt{Request Trans.}) & = & \texttt{PET provider} & r_C(\texttt{Request Trans.}) & = & \{\texttt{Transp. Provider}\} \\
s_C(\texttt{Request Exam.}) & = & \texttt{Transp. Provider} & r_C(\texttt{Request Exam.}) & = & \{\texttt{Hospital}\} \\
s_C(\texttt{Result}) & = & \texttt{Hospital} & r_C(\texttt{Result}) & = & \{\texttt{Transp. Provider}\} \\
s_C(\texttt{Arrival}) & = & \texttt{Transp. Provider} & r_C(\texttt{Arrival}) & = & \{\texttt{PET provider}\} \\
s_C(\texttt{Retransport}) & = & \texttt{PET provider} & r_C(\texttt{Retransport}) & = & \{\texttt{Transp. Provider}\} \\
s_C(\texttt{Return}) & = & \texttt{Transp. Provider} & r_C(\texttt{Return}) & = & \{\texttt{Hospital}\} \\
s_C(\texttt{Confirmation+}) & = & \texttt{PET provider} & r_C(\texttt{Confirmation+}) & = & \{\texttt{Hospital, Transp. Provider}\} \\
s_C(\texttt{Cancel PET}) & = & \texttt{Transp. Provider} & r_C(\texttt{Cancel PET}) & = & \{\texttt{PET provider}\} \\
dom_C(\texttt{Request PET}) & = & D_{Status} & dom_C(\texttt{Confirmation}) & = & D_{Date} \\
dom_C(\texttt{Request Trans.}) & = & D_{Date} & dom_C(\texttt{Request Exam.}) & = & D_\epsilon \\
dom_C(\texttt{Result}) & = & D_{Order} & dom_C(\texttt{Arrival}) & = & D_\epsilon \\
dom_C(\texttt{Retransport}) & = & D_\epsilon & dom_C(\texttt{Return}) & = & D_\epsilon \\
dom_C(\texttt{Confirmation+}) & = & D_{Date} & dom_C(\texttt{Cancel PET}) & = & D_\epsilon
\end{array}
$$

$$
\begin{array}{ll}
\Sigma_{\mathcal{I}} = \{ & (\texttt{Request PET}, uncritical), (\texttt{Request PET}, critical), (\texttt{Result}, abort), (\texttt{Result}, continue), (\texttt{Request Exam.}, \epsilon), (\texttt{Arrival}, \epsilon), \\
& (\texttt{Confirmation}, 1.1.1900), \dots, (\texttt{Confirmation}, 31.12.2099), (\texttt{Confirmation+}, 1.1.1900), \dots, (\texttt{Confirmation+}, 31.12.2099), \\
& (\texttt{Request Trans.}, 1.1.1900), \dots, (\texttt{Request Trans.}, 31.12.2099), (\texttt{Retransport}, \epsilon), (\texttt{Return}, \epsilon), (\texttt{Cancel PET}, \epsilon)\}
\end{array}
$$

XOR-splits G^s_{dx}, the set of event-based XOR-splits G^s_{ex}, and the set of XOR-mergers G^m_x,

- *V is the set of virtual data objects,*
- *$class : I \to C$ assigns a message class to each interaction,*
- *$\to \subseteq (N - E_e) \times (N - \{e_s\})$ is the interaction flow relation,*
- *$\dashrightarrow \subseteq I \times V$ is the data-assignment relation,*
- *$dom_V : V \to \mathcal{D}$ is a function assigning a domain to each virtual data object,*
- *$grd : (\to) \to \mathcal{G}_V$, is a function assigning a guard to each interaction flow.*

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

Definition 4 (Well-Formed DAChor)**.** *A DAChor is well-formed, iff each following property holds:*

- *the start event, each interaction, and each merge node have exactly one successor, i.e., $\forall n \in \{e_s\} \cup I \cup G^m_+ \cup G^m_\times : |\{n' \in N | n \to n'\}| = 1$*
- *each split node has at least one successor, i.e., $\forall g^s \in G^s_+ \cup G^s_{d\times} \cup G^s_{e\times} : |\{n \in N | g^s \to n\}| \geq 1$*
- *each end event, each interaction, and each split node have exactly one predecessor, i.e., $\forall n \in E_e \cup I \cup G^s_+ \cup G^s_{d\times} \cup G^s_{e\times} : |\{n' \in N | n' \to n\}| = 1$*
- *each merge node has at least one predecessor, i.e., $\forall g^m \in G^m_+ \cup G^m_\times : |\{n \in N | n \to g^m\}| \geq 1$*
- *each event-based XOR-split is solely followed by interactions, i.e., $\forall g^s_{ex} \in G^s_{e\times} : \{n \in N | g^s_{ex} \to n\} \subseteq I$*
- *guards of interaction flows are constantly true unless the source of an interaction flow is a data-based XOR-split, i.e., $grd((n_1, n_2)) \neq true \Leftrightarrow n_1 \in G^s_{d\times}$*
- *the data of an interaction is solely assigned to variables of the same data domain, i.e., $\forall i \in I, \forall v \in V : i \dashrightarrow v \;\Rightarrow\; dom_C(class(i)) = dom_V(v)$.*
- *there exists no cycle that solely consists of gateways, i.e., $\nexists g_0, g_1, \dots g_n \in G : g_0 \to g_1 \to \dots \to g_n \to g_0$.*

C. Data-Aware Interaction Net

We introduce the notion of *Data-Aware Interaction Net* (DAI Net). It combines IP Nets [1] and WFD Nets [2]: 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 $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, dom_C, s_C, r_C, \epsilon)$ be an interaction domain. Then: A tuple $\# = (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class, \to, \dashrightarrow, dom_V, grd)$ 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_{fi},*
- *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,*
- *V is the set of variables,*
- *$class : T_I \to C$ is a function assigning a message class to each interaction transition,*
- *$\to \subseteq ((P - P_{fi}) \times T) \cup (T \times (P - \{p_{in}\}))$ is the flow relation,*
- *$\dashrightarrow \subseteq T_I \times V$ is the data-assignment relation. It expresses that the data of an interaction transition is assigned to the related variable,*

Example 2 (DAChor). *Consider the scenario from Fig. 2. Based on its interaction domain $\mathcal{I}$ (cf. Example 1) we can now describe the given scenario as DAChor $DAC = (N, I, G, e_s, E_e, G^s_+, G^m_+, G^s_{d\times}, G^s_{e\times}, G^m_\times, V, class, \rightarrow, \dashrightarrow, dom_V, grd)$:*

$I = \{i_1, \dots, i_8\}, E_e = \{e^1_e, e^2_e\}, G^s_{d\times} = \{g^{s1}_{d\times}, g^{s2}_{d\times}\}, G^m_\times = \{g^m_\times\}, G^s_+ = G^m_+ = G^s_{e\times} = \varnothing, V = \{\mathtt{Status}, \mathtt{Order}\}, \dashrightarrow = \{(i_1, \mathtt{Status}), (i_5, \mathtt{Order})\}$

$\rightarrow = \{(e_s, i_1), (i_1, i_2), (i_2, i_3), (i_3, g^{s1}_{d\times}), (g^{s1}_{d\times}, i_4), (g^{s1}_{d\times}, g^m_\times), (i_4, i_5), (i_5, g^{s2}_{d\times}), (g^{s2}_{d\times}, e^1_e), (g^{s2}_{d\times}, g^m_\times), (g^m_\times, i_6,), (i_6, i_7), (i_7, i_8), (i_8, e^2_e)\}$

$class(i_1) = \mathtt{Request\ PET}$	$class(i_2) = \mathtt{Confirmation}$	$class(i_3) = \mathtt{Request\ Trans.}$	$class(i_4) = \mathtt{Request\ Exam.}$
$class(i_5) = \mathtt{Result}$	$class(i_6) = \mathtt{Arrival}$	$class(i_7) = \mathtt{Retransport}$	$class(i_8) = \mathtt{Return}$

$dom_V(\mathtt{Status}) = D_{status}$	$grd\big((g^{s1}_{d\times}, i_4)\big) = Status = critical$	$grd\big((g^{s1}_{d\times}, g^m_\times)\big) = Status = uncritical$
$dom_V(\mathtt{Order}) = D_{order}$	$grd\big((g^{s2}_{d\times}, e^1_e)\big) = Order = abort$	$grd\big((g^{s2}_{d\times}, g^m_\times)\big) = Order = continue$

- *$dom_V : V \rightarrow \mathcal{D}$ is a function assigning a data domain to each variable,*
- *$grd : T \rightarrow \mathcal{G}_V$, is a function assigning a guard (cf. Def 3) to each interaction flow relation.*

Further, we define

- *$\Sigma_\# := \bigcup_{i \in T_I} \Sigma_{class(i)}$ as the set of all messages corresponding to $\#$*
- *$P^{\rightarrow t} := \{p \in P | p \rightarrow t\}$ as the set of all places preceding t*
- *$P^{\leftarrow t} := \{p \in P | t \rightarrow p\}$ as the set of all places succeeding t*
- *$P^{\leftrightarrow t} := \{p \in P | p \nrightarrow t \ \wedge \ t \nrightarrow p\}$ as the set of the faraway places of t*

Definition 6 (Well-Formed DAI Net). *A DAI Net is well-formed, iff each following property holds:*

- *each transition has at least one preceding 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 data of an interaction transition is solely assigned to variables of the same data domain, i.e., $\forall t_i \in T_I, \forall v \in V : t_i \dashrightarrow v \ \Rightarrow \ dom_C(class(t_i)) = dom_V(v)$.*
- *there exists no cycle solely consisting of places and silent transitions, i.e., $\nexists p_0, p_1, \dots p_n \in P, t_0, t_1, \dots t_n \in T_S : p_0 \rightarrow t_0 \rightarrow p_1 \rightarrow t_1 \rightarrow \dots \rightarrow p_n \rightarrow t_n \rightarrow p_0$.*

D. Mapping DAChors to DAI Nets

In Section III-C, we introduced DAI Nets to define the behavior of DAChors. Thus, we now define a mapping from data-aware choreographies to DAI Nets. This mapping is based on the approach proposed by Decker et al. [1] who define the behavior of iBPMN Choreographies through their transformation to IP Nets.

Definition 7 (Mapping DAChors to DAI Nets). *Let $DAC = (N, I, G, e_s, E_e, G^s_+, G^m_+, G^s_{d\times}, G^s_{e\times}, G^m_\times, V, class, \rightarrow, \dashrightarrow, dom_V, grd)$ be a DAChor (cf. Def. 3). Then, DAC can be mapped to a DAI Net $\#$ that is defined as $\# := (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class', \rightarrow', \dashrightarrow', dom_V, grd')$, whereby*

$$
\begin{array}{lcl}
P & := & \{p_{(n_1,n_2)} | (n_1, n_2) \in \rightarrow \ \wedge \ n_1 \notin G^s_{e\times}\} \\
p_{in} & := & p_{(e_s,n)} \in P, \textit{whereby } e_s \rightarrow n \in N \\
P_{fi} & := & \{p_{(n,e_e)} | p_{(n,e_e)} \in P \ \wedge \ e_e \in E_e\} \subseteq P \\
P_o & := & P - (\{p_{in}\} \cup P_{fi}) \\
T_+ & := & \{t_{g+} | g_+ \in G^s_+ \cup G^m_+\} \\
T^s_\times & := & \{t^s_{(g^s_\times,n)} | g^s_\times \in G^s_d \times \ \wedge \ n \in N \ \wedge \ g^s_\times \rightarrow n\} \\
T^m_\times & := & \{t^m_{(n,g^m_\times)} | g^m_\times \in G^m_\times \ \wedge \ n \in N \ \wedge \ n \rightarrow g^m_\times\} \\
T_I & := & \{t_i | i \in I\}, \ \ T_S := T_+ \cup T^s_\times \cup T^m_\times, \ \ T := T_S \cup T_I \\
class'(t_i) & := & class(i) \\
\rightarrow' & := & \{(p_{(n_1,n_2)}, t_{n_2}) | n_1 \rightarrow n_2 \ \wedge \ n_1 \notin G^s_{e\times} \\
 & & \qquad \wedge \ n_2 \in I \cup G^s_+ \cup G^m_+\} \\
 & \cup & \{(t_{n_1}, p_{(n_1,n_2)}) | n_1 \rightarrow n_2 \ \wedge \ n_1 \in I \cup G^s_+ \cup G^m_+\} \\
 & \cup & \{(p_{(n_1,n_2)}, t^m_{(n_1,n_2)}) | n_1 \rightarrow n_2 \ \wedge \ n_2 \in G^m_\times\} \\
 & \cup & \{(t^m_{(n_0,n_1)}, p_{(n_1,n_2)}) | n_0 \rightarrow n_1 \rightarrow n_2 \ \wedge \ n_1 \in G^m_\times\} \\
 & \cup & \{(p_{(n_1,n_2)}, t^s_{(n_2,n_3)}) | n_1 \rightarrow n_2 \rightarrow n_3 \ \wedge \ n_2 \in G^s_{d\times}\} \\
 & \cup & \{(t^s_{(n_1,n_2)}, p_{(n_1,n_2)}) | n_1 \rightarrow n_2 \ \wedge \ n_1 \in G^s_{d\times}\} \\
 & \cup & \{(p_{(n_0,n_1)}, t_{n_2}) | n_0 \rightarrow n_1 \rightarrow n_2 \ \wedge \ n_1 \in G^s_{e\times}\} \\
\dashrightarrow' & := & \{(t_i, v) | (i, v) \in \dashrightarrow\} \\
grd'(t) & := & \begin{cases} grd\big((g^s_\times, n)\big), & \textit{iff } t = t_{(g^s_\times,n)} \in T^s_\times \\ \textit{true}, & \textit{else} \end{cases}
\end{array}
$$

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

Theorem 1. *Preservation of Well-Formedness*
Let DAC be a DAChor that is mapped to a DAI Net $\#$. If DAC is well-formed, then $\#$ also is well-formed.

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 being placed in the initial place p_{in} and each variable having the empty value ϵ. A marking is called *final*, if all tokens belong to final places of P_{fi}. 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 evaluable and evaluates to *true*.

Definition 8 (DAI Net Markings and Activated Transitions). *Let $\# = (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class, \rightarrow, \dashrightarrow, dom_V, grd)$ be a DAI Net. Then: A marking of $\#$ is a tuple $m = (\odot, val)$ composing two functions with*

- *$\odot : P \rightarrow \mathbb{N}_0$ assigns to each place the number of corresponding tokens,*
- *$val : V \rightarrow \Omega_\mathcal{I}$ assigns to each variable its current value; $val(v)$ is either the empty value ϵ or an element of the variable's domain, i.e., $val(v) \in dom_V(v) \cup \{\epsilon\}$.*

Additionally, for each DAI Net $\#$ we define:

- *the set of all markings $\mathcal{M}_\#$, whereby $\mathcal{M}_\# := \{m = (\odot, val) \mid m \ \text{ is marking of } \# \}$*

Example 3 (Transformation)**.** *The DAChor* $DAC = (N, I, G, e_s, E_e, G^s_+, G^m_+, G^s_{d\times}, G^s_{e\times}, G^m_\times, V, class, \rightarrow, \dashrightarrow, dom_V, grd)$ *from our Example 2 (i.e., the patient transport scenario) is mapped to the DAI Net* $\# = (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class', \rightarrow', \dashrightarrow', dom_V, grd')$ *as follows (cf. Fig. 4):*

$$P = \{p_{(e_s,i_1)}, p_{(i_1,i_2)}, p_{(i_2,i_3)}, p_{(i_3,g^{s1}_{d\times})}, p_{(g^{s1}_{d\times},i_4)}, p_{(g^{s1}_{d\times},g^m_\times)}, p_{(i_4,i_5)}, p_{(i_5,g^{s2}_{d\times})}, p_{(g^{s2}_{d\times},e^1_e)}, p_{(g^{s2}_{d\times},g^m_\times)}, p_{(g^m_\times,i_6,)}, p_{(i_6,i_7)}, p_{(i_7,i_8)}, p_{(i_8,e^2_e)}\}$$

$$p_{in} = p_{(e_s,i_1)} \quad P_{fi} = \{p_{(g^{s2}_{d\times},e^1_e)}, p_{(i_8,e^2_e)}\} \quad P_o = P - (\{p_{in}\} \cup P_{fi})$$

$$T^s_\times = \{t^s_{(g^{s1}_{d\times},i_4)}, t^s_{(g^{s1}_{d\times},g^m_\times)}, t^s_{(g^{s2}_{d\times},e^1_e)}, t^s_{(g^{s2}_{d\times},g^m_\times)}\} \quad T^m_\times = \{t^m_{(g^{s1}_{d\times},g^m_\times)}, t^m_{(g^{s2}_{d\times},g^m_\times)}\} \quad T_+ = \varnothing \quad T_S = T_+ \cup T^s_\times \cup T^m_\times \quad T_I = \{t_{i_1}, t_{i_2}, \ldots, t_{i_8}\}$$

$$V = \{\mathtt{Status}, \mathtt{Order}\} \quad \dashrightarrow' = \{(t_{i_1}, \mathtt{Status}), (t_{i_5}, \mathtt{Order})\}$$

$$\begin{aligned}\rightarrow' = \{&(p_{(e_s,i_1)}, t_{i_1}), (p_{(i_1,i_2)}, t_{i_2}), (p_{(i_2,i_3)}, t_{i_3}), (p_{(g^{s1}_{d\times},i_4)}, t_{i_4}), (p_{(i_4,i_5)}, t_{i_5}), (p_{(g^m_\times,i_6,)}, t_{i_6}), (p_{(i_6,i_7)}, t_{i_7}), (p_{(i_7,i_8)}, t_{i_8}), (t_{i_1}, p_{(i_1,i_2)}),\\ &(t_{i_2}, p_{(i_2,i_3)}), (t_{i_3}, p_{(i_3,g^{s1}_{d\times})}), (t_{i_4}, p_{(i_4,i_5)}), (t_{i_5}, p_{(i_5,g^{s2}_{d\times})}), (t_{i_6}, p_{(i_6,i_7)}), (t_{i_7}, p_{(i_7,i_8)}), (t_{i_8}, p_{(i_8,e^2_e)}), (p_{(g^{s1}_{d\times},g^m_\times)}, t^m_{(g^{s1}_{d\times},g^m_\times)}),\\ &(p_{(g^{s2}_{d\times},g^m_\times)}, t^m_{(g^{s2}_{d\times},g^m_\times)}), (t^m_{(g^{s1}_{d\times},g^m_\times)}, p_{(g^m_\times,i_6,)}), (t^m_{(g^{s2}_{d\times},g^m_\times)}, p_{(g^m_\times,i_6,)}), (p_{(i_3,g^{s1}_{d\times})}, t^s_{(g^{s1}_{d\times},i_4)}), (p_{(i_3,g^{s1}_{d\times})}, t^s_{(g^{s1}_{d\times},g^m_\times)}), (p_{(i_5,g^{s2}_{d\times})}, t^s_{(g^{s2}_{d\times},e^1_e)}),\\ &(p_{(i_5,g^{s2}_{d\times})}, t^s_{(g^{s2}_{d\times},g^m_\times)}), (t^s_{(g^{s1}_{d\times},i_4)}, p_{(g^{s1}_{d\times},i_4)}), (t^s_{(g^{s1}_{d\times},g^m_\times)}, p_{(g^{s1}_{d\times},g^m_\times)}), (t^s_{(g^{s2}_{d\times},e^1_e)}, p_{(g^{s2}_{d\times},e^1_e)}), (t^s_{(g^{s2}_{d\times},g^m_\times)}, p_{(g^{s2}_{d\times},g^m_\times)})\}\end{aligned}$$

$class'(t_{i_1}) = \mathtt{Request\ PET}$	$class'(t_{i_2}) = \mathtt{Confirmation}$	$class'(t_{i_3}) = \mathtt{Request\ Trans.}$	$class'(t_{i_4}) = \mathtt{Request\ Exam.}$
$class'(t_{i_5}) = \mathtt{Result}$	$class'(t_{i_6}) = \mathtt{Arrival}$	$class'(t_{i_7}) = \mathtt{Retransport}$	$class'(t_{i_8}) = \mathtt{Return}$

$dom_V(\mathtt{Status}) = D_{\mathtt{Status}}$	$grd\left(t^s_{(g^{s1}_{d\times},i_4)}\right) = Status = critical$	$grd\left(t^s_{(g^{s1}_{d\times},g^m_\times)}\right) = Status = uncritical$
$dom_V(\mathtt{Order}) = D_{\mathtt{Order}}$	$grd\left(t^s_{(g^{s2}_{d\times},e^1_e)}\right) = Order = abort$	$grd\left(t^s_{(g^{s2}_{d\times},g^m_\times)}\right) = Order = continue$

- *the initial marking* $m^{in}_\# := (\odot_{in}, val_{in}) \in \mathcal{M}_\#$, *whereby*
$$\odot_{in}(p) := \begin{cases} 1, & if\ p = p_{in} \\ 0, & else \end{cases} \quad \wedge \quad \forall v \in V : val_{in}(v) := \epsilon$$
- *the set of all final markings* $\mathcal{F}_\#$, *whereby*
$\mathcal{F}_\# := \{(\odot, val) \in \mathcal{M}_\# | \forall p \in P : \odot(p) \neq 0 \Leftrightarrow p \in P_{fi}\}$

Further, $\rightsquigarrow \subseteq \mathcal{M}_\# \times T$ *is the transition activation relation.* $m \rightsquigarrow t$ *denotes that marking* $m \in \mathcal{M}_\#$ *activates transition* $t \in T$*, iff the following conditions hold:*

1) $\forall p \in P^{\rightarrow t} : \odot(p) \geq 1$,
2) $\forall v \overset{\epsilon}{\sim} grd(t) : val(v) \neq \epsilon$,
3) $grd(t)$ *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 becoming activated. Activated interaction transitions fire, iff 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 + Subsequent Markings of DAI Nets)**.** *Let* $\# = (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class, \rightarrow, \dashrightarrow, dom_V, grd)$ *be a DAI Net, an* $m = (\odot, val), m' = (\odot', val') \in \mathcal{M}_\#$ *be two markings of* $\#$*. Then:*

- $\mathcal{O}_\# := T_S \cup \Sigma_\#$ *is the set of all options on* $\#$*.*
- $opt_\# : \mathcal{M}_\# \rightarrow 2^{\mathcal{O}_\#} : m \mapsto \{o \in \mathcal{O}_\# | \exists m' \ \wedge \ m \overset{o}{\rightarrow} m'\}$ *maps each marking* m *to the options available under* m*.*
- $m \overset{o}{\rightarrow} m'$ *expresses that* m *leads to* m' *by applying option* $o \in opt_\#(m)$ *with:*

Case 1: $o = t_s \in T_S$ *is a silent transition. Then:* $m \overset{t_s}{\rightarrow} m'$ *holds, iff each of the following conditions is met:*

1) $m \rightsquigarrow t_s$,
2) $\forall p \in P^{\rightarrow t_s} : \odot'(p) = \odot(p) - 1$,
3) $\forall p \in P^{\leftarrow t_s} : \odot'(p) = \odot(p) + 1$,
4) $\forall p \in P^{\leftrightarrow t_s} : \odot'(p) = \odot(p)$,

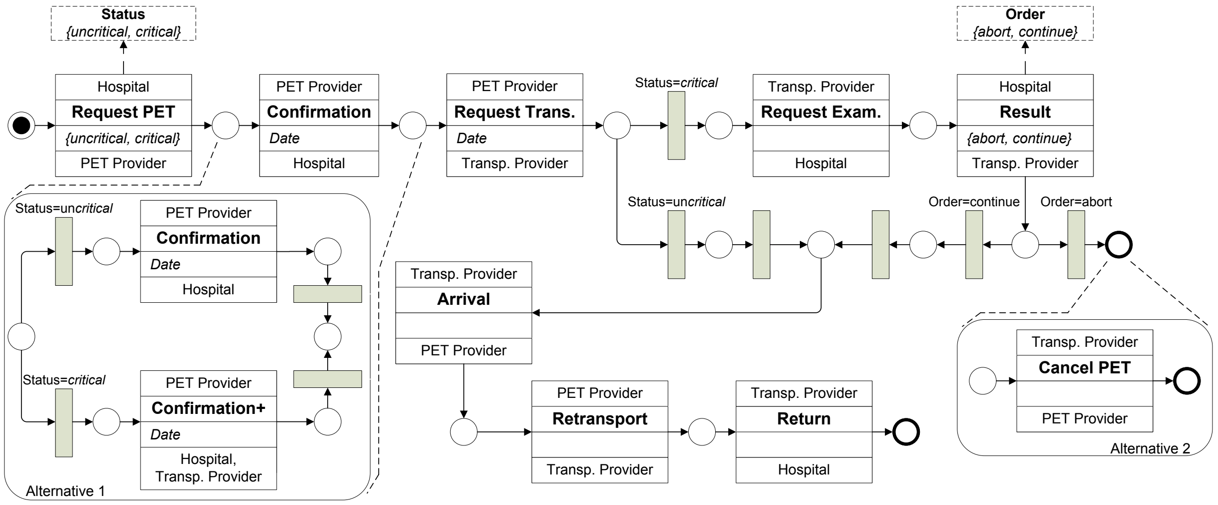

Fig. 4. DAI Net derived for the patient transportation scenario

5) $\forall v \in V : val'(v) = val(v)$.

Case 2: $o = \mu = (c, v) \in \Sigma_{\#}$ is a message. Then: $m \xrightarrow{\mu} m'$ holds, iff each of the following conditions is met:

1) $\forall t_s \in T_S : m \not\rightsquigarrow t_s$, *and*
2) $\exists t_i \in T_I : m \rightsquigarrow t_i \ \wedge \ \mu \in \Sigma_{class(t_i)}$,
3) $\forall p \in P^{\rightarrow t_i} :, \odot'(p) = \odot(p) - 1$,
4) $\forall p \in P^{\leftarrow t_i} : \odot'(p) = \odot(p) + 1$,
5) $\forall p \in P^{\leftrightarrow t_i} : \odot'(p) = \odot(p)$,
6) $\forall v \in V$ *with* $t_i \dashrightarrow v : val'(v) = x$,
7) $\forall v \in V$ *with* $t_i \not\dashrightarrow v : val'(v) = val(v)$.

Based on Def. 9, the following two theorems can be derived (cf. [15]).

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

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

Based on Def. 9, we further 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 trace*.

Definition 10 (Traces, and Prefixes, and Extensions)**.**
(A) *Let $\# = (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class, \rightarrow, \dashrightarrow, dom_V, grd)$ be a DAI Net and $\tau = (\tau_k)_{k\in[1..n]} \in \mathcal{O}^{\star}_{\#}$ be a finite sequence of options (i.e. silent transitions and messages) of length $|\tau| =: n \in \mathbb{N}$. Let further $m = (m_k)_{k\in[1..n+1]} \in \mathcal{M}^{\star}_{\#}$ be a finite sequence of markings of length $n+1$. Then:*

- *$\tau \sim m$ denotes that τ and m are related sequences, iff $\forall l \in [1..n] : m_l \xrightarrow{\tau_l} m_{l+1}$ and $m_1 = m^{in}_{\#}$.*
- *$last : \mathcal{M}^{\star}_{\#} \to \mathcal{M}_{\#}$ with $(m_k)_{k\in[1..n]} \mapsto m_n$ is a function mapping a sequence of markings to its last marking.*
- *$\tau \in \mathcal{O}^{\star}_{\#}$ is a trace, iff $\exists m \in \mathcal{M}^{\star}_{\#}$ and $\tau \sim m$. If $last(m) \in \mathcal{F}_{\#}$, we denote τ a completed trace.*
- *$\mathcal{T}_{\#}$ denotes the set of all traces on $\#$.*
- *$\mathcal{T}^{c}_{\#}$ denotes the set of all completed traces on $\#$.*

(B) *Let $a = (a_k)_{k\in[1..n]}, b = (b_k)_{k\in[1..l]}, c = (c_m)_{m\in[1..m]} \in L \subseteq M^{\star}$ be three elements of set L being a set of sequences over the elements of set M. Then:*

- *$a \trianglelefteq b$ ($a \triangleleft b$) denotes a a prefix (real prefix) of b and b an extension (real extension) of a, iff $n \leq l$ ($n < l$) and $\forall i \in [1..n] : a_i = b_i$,*
- *$a + c = b$ denotes that a is extended by c to b, iff $m + n = l$, and a is prefix of b, and $\forall i \in [1..m] : c_i = b_{n+i}$,*
- *$L^{\trianglelefteq b} := \{a \in L | a \trianglelefteq b\}$ (resp. $L^{\triangleleft b} := \{a \in L | a \triangleleft b\}$) denotes the subset of L that contains all prefixes (resp. real prefixes) of $b \in L$, and*
- *$L^{\trianglerighteq b} := \{a \in L | b \trianglelefteq a\}$ (resp. $L^{\triangleright b} := \{a \in L | b \triangleleft a\}$) denotes the subset of L that contains all extensions (resp. real extensions) of $b \in L$.*

After describing the behavior of a DAI Net by means of its traces, we can 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 lifelocks (cf. [16]).

Definition 11 (Determinism and Soundness)**.**
(A) *We call a DAI Net $\#$ deterministic, iff for each trace τ on $\#$ there exists exactly one related sequence of markings, i.e., $\forall \tau \in \mathcal{T}_{\#} : |\{m \in \mathcal{M}^{\star}_{\#} | m \sim \tau\}| = 1$.*
Let $\#$ be a deterministic DAI Net. Then: Function $mark_{\#}$ maps each trace on $\#$ to its current marking, i.e. the last marking of the related sequence of markings:
$mark_{\#} : \mathcal{T}_{\#} \to \mathcal{M}_{\#} : \tau \mapsto mark_{\#}(\tau) := last(m)$, whereby m is defined by $\tau \sim: m \in \mathcal{M}^{\star}_{\#}$; Since $\#$ is deterministic, the definition of m is unique. Thus, $mark_{\#}$ is well-defined.
(B) *We call a deterministic DAI Net $\#$ sound, iff all following conditions hold:*

- *there exist completed traces on $\#$, i.e., $\mathcal{T}^{c}_{\#} \neq \varnothing$,*
- *each trace on $\#$ is a prefix of a completed trace, i.e., $\forall \upsilon \in \mathcal{T}_{\#} \exists \tau \in \mathcal{T}^{c}_{\#} : \upsilon \trianglelefteq \tau$.*
- *the set of reachable markings is finite, i.e., $|\{m \in \mathcal{M}_{\#} | \exists \tau \in \mathcal{T}_{\#} : last(\tau) = m\}| \in \mathbb{N}$*

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 *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 $\# = (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class, \rightarrow, \dashrightarrow, dom_V, grd)$ be a DAI Net, and $\tau \in \mathcal{T}_{\#}$ be a trace on $\#$. Then:

- *$\Pi_B : A^{\star} \to B^{\star} : a \mapsto \Pi_B(a)$ is the projection function that restricts a sequence $a \in A^{\star}$ to its elements of B,*
- *$\eta \in \Sigma^{\star}_{\#}$ denotes a conversation on $\#$, iff it is the projection of a trace on $\#$ to its messages, i.e., $\exists(\tau) \in \mathcal{T}_{\#} : \Pi_{\Sigma_{\#}}(\tau) = \eta$. η denotes a completed conversation on $\#$, iff it is the projection of a completed trace on $\#$,*
- *$\mathcal{C}_{\#}$ ($\mathcal{C}^{c}_{\#}$) denotes the set of all conversations on $\#$,*
- *$\mathcal{C}^{c}_{\#}$ denotes the set of all completed conversations on $\#$,*
- *$con_{\#} : \mathcal{T}_{\#} \to \mathcal{C}_{\#} : \tau \mapsto con_{\#}(\tau) := \Pi_{\Sigma_{\#}}(\tau)$ 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., *message-determinism*. First, this requires, that firing of silent transitions always terminates (i.e., it is impossible to solely execute silent transitions infinitely). Second, when silent

Example 4 (Traces and Conversations)**.** *Consider the DAI Net $\#$ from Example 3. Its set of completed traces of $\mathcal{T}^c_\#$ consists of traces τ_1, τ_2, and τ_3. Projecting them to their messages leads to the conversations η_1, η_2, and η_3, which build $\mathcal{C}^c_\#$:*

$\tau_1 =<(\mathtt{Request\ PET}, uncritical), (\mathtt{Confirmation}, \_^2), (\mathtt{Request\ Trans.}, \_^2), t^s_{(g^{s1}_{dx}, g^m_{\times})}, t^m_{(g^{s1}_{dx}, g^m_{\times})}, (\mathtt{Arrival}, \epsilon), (\mathtt{Retransport}, \epsilon), (\mathtt{Return}, \epsilon)>$

$\tau_2 =<(\mathtt{Request\ PET}, critical), (\mathtt{Confirmation}, \_^2), (\mathtt{Request\ Trans.}, \_^2), t^s_{(g^{s1}_{dx}, i_4)}, (\mathtt{Request\ Exam.}, \epsilon), (\mathtt{Result}, abort), t^s_{(g^{s2}_{dx}, e^1_e)}>$

$\tau_3 =<(\mathtt{Request\ PET}, critical), (\mathtt{Confirmation}, \_^2), (\mathtt{Request\ Trans.}, \_^2), t^s_{(g^{s1}_{dx}, i_4)}, (\mathtt{Request\ Exam.}, \epsilon), (\mathtt{Result}, continue), t^s_{(g^{s2}_{dx}, g^m_{\times})}, t^m_{(g^{s2}_{dx}, g^m_{\times})},$
$(\mathtt{Arrival}, \epsilon), (\mathtt{Retransport}, \epsilon), (\mathtt{Return}, \epsilon)>$

$\eta_1 = con_\#(\tau_1) := \Pi_{\Sigma_\#}(\tau_1) =<(\mathtt{Request\ PET}, uncritical), (\mathtt{Confirmation}, \_^2), (\mathtt{Request\ Trans.}, \_^2), (\mathtt{Arrival}, \epsilon), (\mathtt{Retransport}, \epsilon), (\mathtt{Return}, \epsilon)>$

$\eta_2 = con_\#(\tau_2) := \Pi_{\Sigma_\#}(\tau_2) =<(\mathtt{Request\ PET}, critical), (\mathtt{Confirmation}, \_^2), (\mathtt{Request\ Trans.}, \_^2), (\mathtt{Request\ Exam.}, \epsilon), (\mathtt{Result}, abort)>$

$\eta_3 = con_\#(\tau_3) := \Pi_{\Sigma_\#}(\tau_3) =<(\mathtt{Request\ PET}, critical), (\mathtt{Confirmation}, \_^2), (\mathtt{Request\ Trans.}, \_^2), (\mathtt{Request\ Exam.}, \epsilon), (\mathtt{Result}, continue),$
$(\mathtt{Arrival}, \epsilon), (\mathtt{Retransport}, \epsilon), (\mathtt{Return}, \epsilon)>$

transitions terminate, the set of activated options must only depend on the messages exchanged before (i.e., it must 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. $\forall \tau \in \mathcal{T}_\#$ holds $\exists N \in \mathbb{N} : \forall \upsilon \in \mathcal{T}^{\trianglerighteq\tau}_\#$ with $|\tau| + N < |\upsilon| \Rightarrow con_\#(\tau) \neq con_\#(\upsilon)$.*

According to Theorem 4 from [15], 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.,*

$$\forall \tau, \upsilon \in \mathcal{T}_\# :$$
$$\big(opt_\#(mark_\#(\tau)), opt_\#(mark_\#(\upsilon)) \subseteq \Sigma_\# \ \wedge\ \Pi_{\Sigma_\#}(\tau) = \Pi_{\Sigma_\#}(\upsilon)\big) \ \Rightarrow$$
$$opt_\#(mark_\#(\tau)) = opt_\#(mark_\#(\upsilon))$$

Let $\#$ be a deterministic, sound and message-deterministic DAI Net. Then: Function $mark_\#$ maps each conversation to the set of messages it activates:
$mo_\# : \mathcal{C}_\# \to 2^{\Sigma_\#} : \eta \mapsto mo_\#(\eta) := opt_\#(mark_\#(\tau))$, $\tau \in \mathcal{O}^\star_\#$ is defined by $\eta = con_\#(\tau)$ and $opt_\#(mark_\#(\tau)) \subseteq \Sigma_\#$. Since $\#$ is message-deterministic, the definition is unique. Thus, $mo_\#$ is well-defined.

Until now, we solely considered DAI Nets and conversations from a global perspective. However, a role solely knows messages of a conversation it sends or receives itself. Thus, in Def. 14 the view of a role on 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 $\mathcal{I} = (\mathcal{R}, \mathcal{D}, C, dom_C, s_C, r_C, \epsilon)$ be an interaction domain and let the tuple $\# = (P, p_{in}, P_o, P_{fi}, T, T_S, T_I, V, class, \to, \dashrightarrow, dom_V, grd)$ be a sound, deterministic, and message-deterministic DAI Net and let $R \in \mathcal{R}$ be a role. Then we can define the following views

- *$vc^R_\# : \mathcal{C}^\star_\# \to \Sigma^\star_R : (\eta_k)_{k\in[1..n]} \mapsto vc^R_\#(\eta) := \Pi_{\Sigma_R}(\eta)$ maps each conversation on $\#$ to the view of R on it, wherby the view is the projection to the messages sent or received by R,*
- *$vc^{R\mapsto}_\# : \mathcal{C}^\star_\# \to \Sigma^\star_R : (\eta_k)_{k\in[1..n]} \mapsto vc^{R\mapsto}_\#(\eta) := \Pi_{\Sigma_{R\mapsto}}(\eta)$ maps each conversation on $\#$ to the messages sent by R,*
- *$vo^R_\# : 2^{\Sigma_\#} \to 2^{\Sigma_R} : M \mapsto vo^R_\#(M) := M \cap \Sigma_R$ maps each set of messages to its messages sent or received by R,*
- *$vo^{R\mapsto}_\# : 2^{\Sigma_\#} \to 2^{\Sigma_{R\mapsto}} : M \mapsto vo^{R\mapsto}_\#(M) := M \cap \Sigma_{R\mapsto}$ maps each set of messages to its messages sent by R and,*

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 $R \in \mathcal{R}$:*

- *$\#$ is realizable, iff the messages role R may send solely depend on the messages, R has sent and received before, i.e., $\forall R \in \mathcal{R} : \forall \eta, \kappa \in \mathcal{C}_\# : vc^R_\#(\eta) = vc^R_\#(\kappa) \Rightarrow vo^{R\mapsto}_\#(mo_\#(\eta)) = vo^{R\mapsto}_\#(mo_\#(\kappa))$*
- *$\#$ clearly terminates, iff it solely depends on the messages R has sent and received before, whether or not further interaction with R will occur, i.e.,*
  $\forall R \in \mathcal{R} : \forall \eta \in \mathcal{C}^c_\# \nexists \kappa \in \mathcal{C}_\# : vc^R_\#(\eta) \lhd vc^R_\#(\kappa)$

An important issue concerns decidability of the introduced properties of DAI Nets and DAChors; i.e., determinism, soundness (cf. Def. 11), message-determinism (cf. Def. 13), and realizability and clear termination (cf. Def. 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 [2], [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 [5] and BPEL4Chor [6]. 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 [7], [13], [14], [19]–[22]. Some

[1]For reasons of simplification, we abstract from irrelevant message content in Example 4

approaches use data dependencies to interconnect processes and to define process interaction [23], [24]. Examples of interaction models (i.e., the paradigm we apply) include Service Interaction Pattern [9], WSCDL [10], iBPMN Choreographies [1], and BPMN 2.0 Choreographies [5]. Our approach has been mainly inspired by [1], which defines the behavior of iBPMN Choreographies through their transformation to Interaction Petri Nets, and further discusses correctness and realizability. Also, realizability of interaction models is discussed in [11], [25]. Furthermore, [12] 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 that are aware of the content of messages. 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 because 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 bottom 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 varity 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 bahavior of DAI Nets is definde. 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 C^3Pro funded by the German Research Foundation (DFG), Project number: RE 1402/2-1.

References

[1] G. Decker and M. Weske, "Interaction-centric modeling of process choreographies," *Information Systems*, vol. 36, no. 2, pp. 292–312, 2011.

[2] N. Trčka, W. M. P. van der Aalst, and N. Sidorova, "Data-flow anti-patterns: Discovering data-flow errors in workflows," *Proc. CAiSE'09*, pp. 425–439, 2009.

[3] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems*. Springer, 2012.

[4] M. Reichert, T. Bauer, and P. Dadam, "Enterprise-wide and cross-enterprise workflow management: Challenges and research issues for adaptive workflows," in *Proc. Enterprise-wide and cross-enterprise workflow management*, 1999, pp. 55–64.

[5] OMG/BPMI, "Bpmn version 2.0," 2011.

[6] G. Decker and et al., "Bpel4chor: Extending bpel for modeling choreographies," *ICWS 2007*, pp. 296–303, 2007.

[7] W. M. P. van der Aalst and et al., "Multiparty contracts: Agreeing and implementing interorganizational processes," *The Computer Journal*, vol. 53, no. 1, pp. 90–106, 2010.

[8] G. Decker and A. P. Barros, "Interaction modeling using bpmn," in *Business Process Management Workshops*, 2007, pp. 208–219.

[9] A. Barros, M. Dumas, and A. Ter Hofstede, "Service interaction patterns," *Business Process Management*, pp. 302–318, 2005.

[10] W3C, "Web services choreography description language v1.0," 2005.

[11] N. Lohmann and K. Wolf, "Realizability is controllability," *Web Services and Formal Methods*, pp. 110–127, 2010.

[12] P. Poizat and G. Salaün, "Checking the realizability of BPMN 2.0 choreographies," in *Proc. SAC'11*, 2011.

[13] D. Yellin and R. Strom, "Protocol specifications and component adaptors," *ACM TOPLAS*, vol. 19, no. 2, pp. 292–333, 1997.

[14] S. Rinderle, A. Wombacher, and M. Reichert, "Evolution of process choreographies in DYCHOR," in *Proc. CoopIS'06*, 2006, pp. 273–290.

[15] D. Knuplesch and M. Reichert, "A formal framework for data-aware process interaction models," Ulm University, Tech. Rep. 2012-06, 2012.

[16] W. M. P. van der Aalst, "Verification of workflow nets," *Application and Theory of Petri Nets*, pp. 407–426, 1997.

[17] M. Reichert and P. Dadam, "ADEPT$_{flex}$ – supporting dynamic changes of workflows without losing control," *Journal of Intelligent Information Systems*, vol. 10, no. 2, pp. 93–129, 1998.

[18] O. Kopp and F. Leymann, "Do we need internal behavior in choreography models," in *Proc. ZEUS'09*, 2009, pp. 2–3.

[19] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "An integrated workbench for model-based engineering of service compositions," *Services Computing Transactions on*, vol. 3, no. 2, pp. 131–144, 2010.

[20] S. Rinderle, A. Wombacher, and M. Reichert, "On the controlled evolution of process choreographies," in *Proc. ICDE'06*, 2006, p. 124.

[21] M. Reichert, T. Bauer, and P. Dadam, "Flexibility for distributed workflows," in *Handbook of Research on Complex Dynamic Process Management*. BSR, IGI Global, 2009, pp. 137–171.

[22] M. Reichert and T. Bauer, "Supporting ad-hoc changes in distributed workflow management systems." in *Proc. CoopIS'07*, 2007, pp. 150–168.

[23] V. Künzle and M. Reichert, "Philharmonicflows: towards a framework for object-aware process management," *Software Maintenance and Evolution: Research and Practice*, vol. 23, no. 4, pp. 205–244, 2011.

[24] D. Müller, M. Reichert, and J. Herbst, "Data-driven modeling and coordination of large process structures," in *Proc. CoopIS'07*, 2007, pp. 131–149.

[25] G. Decker, "Realizability of interaction models," in *Proc. ZEUS'09*, 2009, pp. 55–60.

[26] X. Fu, T. Bultan, and J. Su, "Conversation protocols: A formalism for specification and verification of reactive electronic services," in *Proc. CIAA'04*, 2004, pp. 188–200.

[27] ——, "Realizability of conversation protocols with message contents," *Web Services Research*, vol. 2, no. 4, pp. 68–93, 2005.

[28] D. Knuplesch, L. Ly, S. Rinderle-Ma, H. Pfeifer, and P. Dadam, "On enabling data-aware compliance checking of business processmodels," in *Proc. ER'10*, 2010, pp. 332–346.