Distributed and Parallel Databases (2019) 37:651–695
https://doi.org/10.1007/s10619-018-7251-3

Optimizing data-flow implementations for inter-organizational processes

Julius Köpke[1] · **Marco Franceschetti**[1] · **Johann Eder**[1]

Published online: 10 October 2018

Abstract
Inter-organizational business processes can be designed top-down: starting from a global process model, where each activity and gateway is assigned to one of the participating organizations, local processes (or views/stubs for refinement) are generated for each organization. The resulting inter-organizational process is executed by the local processes in a fully distributed manner. Implementing the data-flow in such a scenario is highly complex, both because of the combinatorial explosion of possible solutions and because of conflicting goals for implementing the data-flow. We present a flexible, heuristic algorithm for implementing the data-flow for inter-organizational processes according to various user preferences, which significantly improves over the base-line approaches proposed so far. The approach is fully implemented and a comprehensive, multi-dimensional evaluation is provided.

Keywords Inter-organizational business processes · Data-flow implementation · Inter-organizational workflows · Process partitioning · Choreography generation

1 Introduction

Inter-organizational business processes [1–11] organize the collaboration of different parties to achieve business objectives. Inter-organizational peer to peer processes, which we study here, are considerably different from *intra*-organizational processes

✉ Julius Köpke
julius.koepke@aau.at

Marco Franceschetti
marco.franceschetti@aau.at

Johann Eder
johann.eder@aau.at

[1] Department of Informatics Systems, Alpen-Adria-Universität Klagenfurt, Universitätsstrae 65, 9020 Klagenfurt, Austria

as the parties are independent and autonomous, only bound to executing the process by contracts, resp. by their interests.

Technically, we can see an inter-organizational process as an ensemble of communicating processes (called local processes) of the different parties. There are different ways to represent and develop inter-organizational processes. One way is to define the interaction between pairs of local processes as protocols. The inter-organizational (or global) process is then an emergent phenomenon - neither directly designed nor controllable. A second approach is that one of the parties dominates (such as in some supply chain processes) and controls and enacts the inter-organizational process. Here the challenges are similar to intra-organizational processes spanning different organizational units. The more general p2p approach to inter-organizational processes, which we follow here, describes an inter-organizational process by means of a process model, which defines as usual the steps (activities and control structures) of a process and assigns each step to one of the partners. The activities, thereby, can be abstract or generic, corresponding to a part of the partners local process. Internals of these local processes are not exposed to the other parties. This general approach has been studied in the past with several approaches such as [3,7,12–16].

Parties in inter-organizational processes are considered independent and autonomous, hence we cannot assume a central coordination of the process. Therefore, the thread of control has to be passed between the parties by explicit exchange of messages. In earlier papers we developed procedures for generating these interaction steps to realize the control-flow between different parties. Therefore, instead of assuming a choreography (e.g. in form of BPMN choreography diagrams [17] or BPMN collaboration diagrams [17]) as input, our aim is to automatically derive choreographies from a global process model.

In this paper we focus on the data-flow. Previous approaches concentrating on the control-flow ignore the data-flow such as [3,7,13], assume central databases such as [12], or apply static straightforward strategies for implementing the data-flow via message exchanges [14,15]. It is our intention to show that there are huge differences in the implementations with respect to some quality criteria, and propose an algorithm which generates good implementations of data-flow with reasonable scalability.

We assume that part of the process description is the definition of the data requirements and the data production of the steps of a process, this means a definition of input and output parameters of each activity. In local processes the management of the data logistics between activities is done by the process enactment service or by access to a common data store. If such a common data store (distributed database, shared cloud services, joint central database, etc.) exists, then the parties can retrieve up-to-date data items from this data store whenever they are required and write the updated values to that data store in the same way as activities read from and write to an enterprise database in *intra*-organizational processes. Correctness of the data can be ensured by traditional transaction mechanisms. In inter-organizational processes (e.g. complex international trade) the existence of such a common data store cannot be expected and, therefore, we assume here that such a common data store, which can be accessed by all parties, does not exist. We also have to assume that there is no central process enactment service but the process is enacted by local enactment services and the exchange of messages between local enactment services according to the process

definition. Therefore, there is no enactment service which can take the role of the data distributor. As a consequence, for the types of true p2p - processes we consider here also the data-flow between the steps of the global process has to be realized by exchanging messages between parties. The major challenge is now to inject messages in the process definition to realize the required data-flow.

In [18], we studied the problem of where to inject which data transferring messages into the definition of an inter-organizational process to derive an 'optimal' implementation. We showed that starting from a correct implementation of the data-flow we can derive any other correct implementation by a sound and complete set of equivalence transformations. We also argued that there are different objectives to follow for choosing between correct implementations. In principle, this allows to derive the best fitting data-flow implementation using custom objective functions. However, a direct implementation of this approach with heuristic search functions is not feasible for reasonably sized processes, because of the complexity of the problem. In [19] we defined measures for technical qualities of different implementations. These qualities cover the number of interactions or messages, the number of data transmissions and the impact on the complexity of some implementation. We showed how simple implementation strategies (called base-line approaches) optimize one of the measures at the expense of the other measures.

Here we present and evaluate a heuristic algorithm to generate a good data-flow implementation, where *good* is defined by measures expressing α the number of necessary messages, β the amount of transmitted data, and γ the increase in collaboration complexity for implementing the data-flow.

The contributions of this paper are the following:

1. We extend and formalize the measures describing qualities of data-flow implementations and discuss implementation classes.
2. We introduce a customizable heuristic algorithm for the generation of optimized data-flow implementations for different user-requirements.
3. We provide a comprehensive multi-dimensional evaluation of our approach. We have shown that our algorithm achieves on average significantly better measure scores depending on user requirements and we demonstrate that the approach is sufficiently scalable to be feasible for large inter-organizational processes.

The remainder of this paper is structured as follows. In Sect. 2, we motivate the problem of implementing inter-organizational data-flow via message exchanges in an example, show that different implementations are possible, discuss the pros and cons of a set of possible implementations. Section 3 formalizes the underlying process model. Section 4 refines measures for assessing data-flow implementations from [19]. Section 5 newly introduces the class of non-redundant augmentations and revisits valuable other classes of data-flow implementations previously proposed in [19]. Section 6 discusses optimization strategies for transitive submissions, lays the formal ground for incrementally computing augmentations and finally introduces a heuristic approach for deriving optimized data-flow implementations for various augmentation classes. Section 7 presents a comprehensive evaluation of our approach against existing base-line strategies. Finally, Sect. 8 discusses related work and Sect. 9 concludes the paper.

2 Preliminaries and motivating example

Our goal is to automatically derive local process models for each organization (actor or partner) participating in an inter-organizational process defined by a global process model. Each such local process model is a view on the actual local process and may be extended and refined, e.g. activities of the global process model may be realized by sub-processes in local process models. The inter-organizational process is then executed in a fully distributed manner by the interplay of the local processes exchanging messages. Since we do not assume a global data store, data-flows crossing organizational borders need to be realized in form of message exchanges.

Our approach consists of 4 phases:

1. Design of a global process model including data-flow specifications of tasks and gateways in form of read and write accesses to case variables.
2. Assignment of actors (organizations) to activities and gateways.
3. Insertion of communication steps into the global process model for the implementation of inter-organizational data-flow (called augmentation of the global process model).
4. Automatic generation of the messages for distributing the control-flow and deriving local process models.

In this paper, we focus on phase (3) generating augmentations for implementing the data-flow. Procedures for phase (4), the insertion of messages to implement the distributed control-flow and the generation of local process models, were already proposed in our previous work [13] and are not repeated here.

Augmentations realize the implicit data-flow of the global process model via explicit message exchanges, if organizational boundaries are crossed. E.g. if an activity B_b of actor b reads a variable y that was modified by a previous activity A_a executed by an actor a, the value written by A_a needs to be transmitted to b prior to executing B_b. Message exchanges are included in the augmentation in form of communication steps. A communication step (a, b, c, X) specifies that a data item X is transmitted from a sender a to some recipient b if condition c holds. Communication steps, however, abstract from the communication mode (push/pull, synchronous/asynchronous).

2.1 Example

Figure 1 shows an example of a global inter-organizational process where actors (organizations) are assigned for each task and gateway. Clearly, actor assignment is a design-time definition referring to types of actors rather than concrete entities. We denote the actor (organization) in charge for executing a task or a gateway in square brackets after the node label. The inter-organizational global process is executed by a general practitioner GP, a diagnosis institute DI, a rehabilitation specialist R, an insurance company I and an accounting office AO. First a general practitioner GP requests additional examinations. Then two streams of actions are performed in parallel (any order): a magnetic resonance tomography (MRT) examination is performed by a diagnosis institute DI in task $A2$ and an assessment of the fitness of the patient is performed by a rehabilitation specialist R in task $A4$. After each examination an

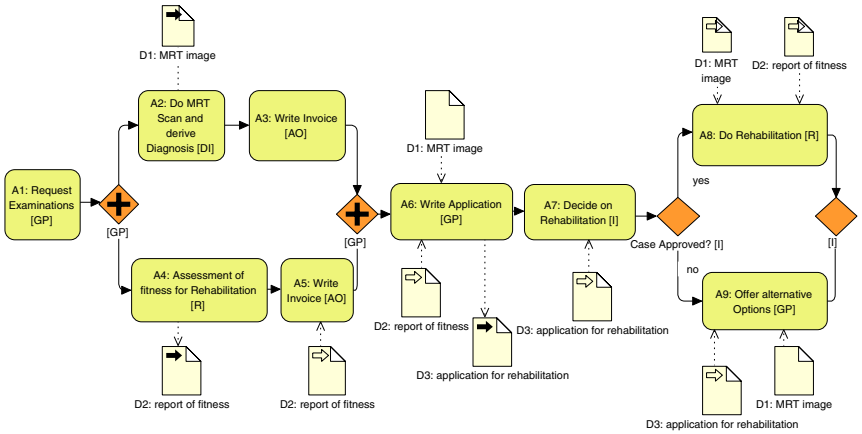

Fig. 1 Example process

invoice is created by an accounting office AO in tasks $A3$ and $A5$. Following the par-join, the general practitioner GP writes an application for a rehabilitation in task $A6$. Next, the insurance company I decides on the application in task $A7$. If the case is approved, the rehabilitation is started by the rehabilitation specialist R in task $A8$. Otherwise the general practitioner GP discusses alternative options to the patient in task $A9$.

In the example, three data objects ($D1$ to $D3$) are modeled. $D1$ are MRT images created by $A2$. It is potentially a very large data object. $D2$ is a report of the fitness derived by the rehabilitation specialist in task $A4$. $D3$ is an application for rehabilitation created by the general practitioner in task $A6$. Other data objects such as invoices are not modeled since they are not relevant for the future steps of the process or they might be sent as physical documents and are therefore out of scope of this paper, where only electronic data-exchange is addressed. The input and output data of tasks from Fig. 1 implicitly encodes the data-flow between the tasks. In the example, we use the usual BPMN notation for input (non black arrow) and output (black arrow) documents. We only need to address data-flow that crosses organizational boundaries. All such data-flows need to be realized via messages. We show the required cross-organizational data-flows (DF1 to DF5) in the upper part of Fig. 2 as dotted lines from the source or producer (last activity writing to some variable) to each task that potentially requires the data value.

2.2 Base-line approaches

We have presented and evaluated existing approaches for the implementation of cross-organizational data-flows via message exchanges in [19].

Figure 2 shows 4 different augmentations for the example process in Fig. 1. We represent communication steps similar to BPMN choreography tasks. They are represented as rounded boxes, where the origin of the data transmissions is denoted on the top of the box and the receiving partner is denoted on the bottom of the box. The

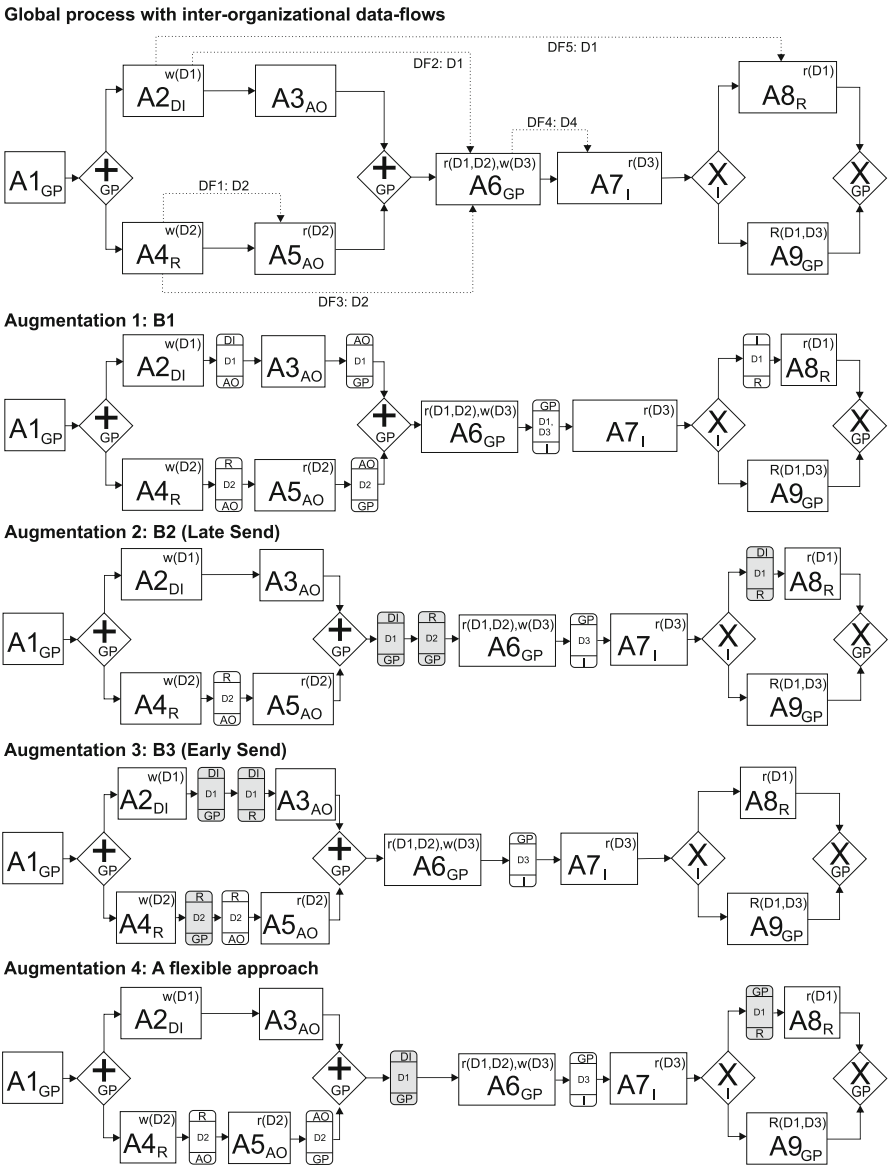

Fig. 2 Example process from Fig. 1 with cross-organizational data-flow (dotted lines)

transmitted variables (data objects) are denoted in the middle. We now briefly discuss each approach based on the example.

B1: data-flow follows control-flow In the data-flow follows control-flow approach, communication steps are restricted to existing control-flow edges between different participants of the global process. If there is an edge between 2 activities assigned to different partners, the implementation of the control-flow will generate a message

from the partner executing the first activity to the partner executing the second activity. Therefore, a communication step sending a message from participant a to participant b can only be placed between two nodes A_a, B_b executed by a and b if there is a control-flow edge (A_a, B_b) in the global process. This strategy (Base-line $B1$) is implicitly assumed in approaches that abstract from the data-flow perspective [3,7,13].

Augmentation 1 in Fig. 2 follows this approach. It uses 6 communication steps. Trivially, all of them can be included in control-flow messages. Therefore, no additional interactions are required for implementing the data-flow. The augmentation contains 7 transmissions of data elements (variables), of which 6 are executed unconditionally, one (in the xor-block) is executed conditionally. It can be seen as an advantage, that no additional interactions are required. However, Augmentation 1 transmits the MRT image $D1$ via the accounting office AO and the insurance company I to the rehabilitation specialist R. This causes multiple transmissions of potentially large data objects compared to direct transmissions. Even more importantly, there may be confidentiality issues since neither the accounting office nor the insurance company execute any step that requires the MRT images.

B2: direct late send Data is sent from the activity that last updated the data element (origin) to the activity requiring the data at the latest possible point in time. Following this approach, a data-flow message can only be included in a control-flow message, if the task requiring the data is a direct successor of the origin task in the global process. A variant of this base-line approach is applied in [15].

Augmentation 2 in Fig. 2 shows an augmentation following this approach. It requires a total of 5 communication steps. One communication step is conditionally executed, three communication steps require additional message exchanges, because the payload cannot be integrated into control-flow messages. We depict these steps in grey. Notably, in Augmentation 2 $D1$ will only be transmitted to the rehabilitation specialist, if the rehabilitation is granted by the insurance company. Therefore, there are no confidentiality issues. However, the local process of the diagnosis institute gets more complex. In Augmentation 1, the diagnosis institute only needs to know that it is called by the general practitioner and that after doing the examination, data needs to be forwarded to the accounting office. When 'late send' is implemented, the diagnosis institute becomes actually part of the xor-block. This also means that the xor-block needs to be included in the local process of the diagnosis institute.

B3: direct early send In this approach, whenever a producer writes some data item, it is immediately distributed to each partner potentially needing this data item. This base-line approach is proposed in [20,21].

Augmentation 3 of Fig. 2 follows this strategy. The solution uses 5 unconditional communication steps. Three communication steps require additional interactions as they cannot be integrated into control-flow messages. Most notably, the transmission of the MRT images to the rehabilitation specialist is performed unconditionally. Therefore, even if the rehabilitation is not granted by the insurance company, the rehabilitation specialist will receive the MRT images. This adds not needed data transmissions at runtime and also imposes confidentiality concerns. On the positive side, the diagnosis institute does not need to know about later decisions (as for $B2$) and it does not get involved in the xor-block as it was the case for $B2$.

2.3 Discussion

We discuss the base-line approaches in three aspects: (1) the number of additional message exchanges generated for implementing the data-flow, (2) the number of transmissions of data items (corresponding to bandwidth requirements), and (3) the additional complexity of the local processes. The base-line approaches $B1$ and $B2$ have quite distinct strengths and weaknesses. $B1$ requires no additional messages for implementing the data-flow. Therefore it does not require additional interactions and thus has no impact on the coupling between participants. Additionally it does not increase the complexity of the resulting local processes. However, data may be transmitted over and over again leading to high amounts of transmissions and to confidentiality concerns. In contrast, $B2$ transmits data only if it is certainly needed by the recipient. As a result it provides minimal bandwidth requirements and does not impose any confidentiality concerns. However, communication steps can only be piggy-packed with the control-flow in specific cases. Therefore, additional interactions are required for implementing the data-flow which also increases the coupling between participants as well as the complexity of local processes since decisions in the control-flow need to be respected for realizing the cross-organizational data-flow. $B3$ has the same behavior as $B2$ in terms of additional interactions, as the data-flow cannot always be included in the control-flow. It requires less data transmissions than $B1$ but more than $B2$ since all participants who may potentially need some data will receive it. Regarding additional complexity of local processes, $B3$ performs better in our example as future decisions have no impact on the sender. However, in case of activities that write to data variables conditionally, similar effects occur: the receiver may need to know about earlier decisions and may need to include additional xor-gateways in its local process.

In summary, the base-line approaches typically are good in one aspect, but at the cost of the other aspects. The research question for us was now: can we find a better solution? Augmentation 4 in Fig. 2 does not strictly follow one of the base-lines. It requires a total number of 5 communication steps, where one is conditional. Three communication steps can be included in the control-flow. Most notably, by transmitting $D2$ from the accounting office to the general practitioner, one additional transmission can be included in the control-flow compared to Augmentation 2. This does not impose confidentiality concerns or additional data transmissions because the accounting office also needs $D2$. Following $B2$, always the participant, who last updated some data object, has to send it to each consumer. This requires the diagnosis institute to become part of the xor-block in Augmentation 2. We avoid this problem in Augmentation 4 by sending $D1$ from the general practitioner to the rehabilitation specialist. This provides an overall advantage since the general practitioner is already part of the xor-block and she requires the MRT images anyhow.

Our approach for improvement starts with generating augmentations like Augmentation 4 minimizing the number of additional messages and the amount of transmitted variables, while reducing the negative impact on the complexity of the local processes.

Notably, all augmentations in the example contain only required communication steps, i.e. removing any communication step would result in an augmentation that does not correctly implement the implicit data-flow of the global process. As a counter-

example a simpler implementation of $B1$ would forward all data objects via every control-flow edge of the global process and only apply specific rules at parallel-join gateways. Such a solution is trivially correct but contains redundant data transmissions. In this paper we are interested in finding augmentations that are non-redundant.

2.4 On finding 'optimal' implementations

The base-line approaches shown above apply static strategies that may result in suboptimal solutions. A possibility to automatically obtain optimal data-flow implementations is to apply an exhaustive search employing some objective function to select the best solution for the given user requirements. All potential correct solutions can be generated with the equivalence transformations presented in [18] and the objective function can be composed of the measures for augmentations proposed in [19]. However, this approach does not scale, since the size of the search space explodes with the number of actors and with the number of xor-gateways in the process. Preliminary experiments showed that this approach does not even scale for comparatively small process models. We now informally motivate why an exhaustive search is not feasible.

Given a sequential process containing some activity $A1_{P1}$ writing to a variable x and some later executed activity An_{Pn} reading the value of x produced by $A1_{P1}$. A communication step transmitting x directly from $P1$ to Pn can be placed at any position between $A1_{P1}$ and An_{Pn}. Having 10 nodes of different actors in-between this results in 9 different solutions. Additionally, the data may be routed indirectly over any subset of the other actors in any sort order. This results in an exponential growth by the number of actors. For each such subset of actors all permutations of actors are possible. Assuming m actors in one such subset of actors already results in $m!$ solutions. Each such transitive transmission may be implemented with steps distributed over any of the positions between $A1_{P1}$ and An_{Pn}. Additionally, process models are not limited to sequences. When there are xor-gateways between $A1_{P1}$ and An_{Pn}, then this already exponentially large number of potential solutions grows again exponentially by the number of xor-gateways between $A1_{P1}$ and An_{Pn}.

Since a search for an optimal solution for arbitrary objective functions is not feasible, we provide a heuristic algorithm for finding good solutions for various user requirements.

2.5 Research goals, assumptions, and limitations

The goal of the research reported here was to develop an algorithm for the generation of optimal implementations of the data-flow for inter-organizational p2p processes described in simple process models.

The following assumptions and limitations apply:

- The processes are peer2peer processes without any central coordination.
- There is no joint data store, the processes act on a 'shared nothing' architecture.
- The processes are defined in a simple process model consisting of activities connected with the core set of control patterns and augmented with actors and data parameters.

– The objective for the optimization is to minimize the amount of required messages and data transfers without unduly increase the complexity of the local processes.

All the rest of the paper should be considered with these assumptions and limitations. In particular, when we say 'better' or 'optimal' this is always with respect to the objective stated above. The assumption of a 'shared nothing' architecture follows the observation that many parties can be involved in an inter-organizational business process and that the requirements for participating in an inter-organizational process should be as low as possible. We cannot think of lower assumptions than being able to receive and send messages.

3 Process model

3.1 Global process model

We consider here an inter-organizational process model[1] based on the standard minimum workflow control patterns [22]: full blocked acyclic workflow nets. In a nutshell: the process model consists of nodes and edges, where the nodes are activities, communication steps, or control steps (and-split/-join, xor-split/join), and the edges express a precedence relationship [22,23]. We chose block-structured workflow nets to avoid the typical problems of unstructured business processes that manipulate data [24] which are even more severe in an inter-organizational setting. We present the process model from [18,19] here to be self-contained but we newly extend it with *labels* to provide a formal concise basis for the following sections.

Nodes are assigned to the parties of an inter-organizational collaboration for execution. Activities read and/or write variables. xor-split nodes read boolean decision variables. Communication steps are used for the transfer of control and for the transmission of data (variables) between parties. We first introduce the global process model, where data is exchanged via variables and then present an augmented process model where the implicit data-flow between different actors (organizations) is implemented via communication steps.

There are basically two possibilities for representing and evaluating the condition of an xor-split in inter-organizational process models [18]: (1) The condition is not defined in the global process, or (2) the condition is defined using some global variables. In case (1), the participant who is in charge of the xor-split takes the decision (probably involving local variables) and informs the other parties of the result, if necessary. In case (2) each participant receives all variables appearing in the condition to make the decision (i.e. evaluate the condition) locally. This may result in additional communication overhead. For simplicity and clarity, we follow a combination of (1) and (2): each xor-split has a unique boolean decision variable. One of the actors is in charge to compute this decision variable and communicate it to the other parties, which need to know it so that they can execute the xor-split locally. This strategy

[1] For being self-contained, we repeat the process model from [19]. As an extension to [19], we introduce labels to provide a more concise basis for the following sections.

also conforms to the recent OMG Decision Model and Notation DMN, where specific decision tasks are used to evaluate decisions prior to conditional gateways.

Parallel execution may raise race conditions, which in a non-distributed setting can be solved by using a transactional data store. Since for inter-organizational processes we do not assume that all parties have access to a distributed transactional data store, we do not allow parallel read-write and write-write dependencies between variables. Consequently, data-flow dependencies may only exist between activities connected by control-flow dependencies expressed in the edges of the process model.

Definition 1 (*Global process model*) A *global process model* $P = (N, E, V, A)$ with a set of nodes N connected by a set of directed edges E with set of variables V and a set of parties (actors) A forms a directed acyclic graph. For each node n we define $n.type$ the type *(activity | xor-split | xor-join | and-split | and-join)*, $n.name$ the name of the node, $n.v^r \subseteq V$, the set of variables read and $n.v^w \subseteq V$, the set of variables written, and $n.a \in A$, the actor executing the node.

Each edge $(m, n) \in E, m, n \in N$, describes a precedence constraint between nodes m and n. There is one node without predecessor, called start node and one node without successor called stop node. $n.succ = \{m|(n, m) \in E\}$ and $n.pred = \{m|(m, n) \in E\}$ represent the successors, resp. predecessors of a node. $n.succ+$ and $n.pred+$ represent the transitive closure of the successor and predecessor relation. In order to access successors or predecessors on a specific sub graph sg of P, we write $n.succ_{sg}+$ or $n.pred_{sg}+$. xor-split nodes and and-split nodes have exactly 2 successors, xor-join and and-join have exactly 2 predecessors.

$D \subset V$ is a set of boolean decision variables. Each xor-split node x is associated with a unique decision variable $d \in D$. One of the outgoing edges of an xor-split is adorned with d, the other with $\neg d$.

The process model is full blocked, i.e. each split node is associated with exactly 1 join node such that each path originating in the split node to the end node includes the associated join node. □

In the remainder of the paper we use the usual object-style dot notation to access components of a process model. E.g. for a process $P = (N, E, V, A)$, we address the set of edges via $P.E$.

3.2 Labels

We interpret the decision variables as propositional letters. Let $d \in D$ be a decision variable. The literal d represents that the decision variable is true and $\neg d$ that it is false. A minterm over d is a conjunction of literals which contains for each $d \in D$ either d or $\neg d$.

We use propositional labels for nodes to indicate through which path a node can be reached. We define a label as a conjunction of decisions which lead to the considered node.

Definition 2 (*Propositional terms*) Let Δ be a set of propositional letters, let $\wedge$, $\vee$, $\neg$, $\rightarrow$ be the propositional conjunction, disjunction, negation and implication, respectively. Let $\delta \in \Delta$: then δ and $\neg\delta$ are literals, $\bar{\Delta}$ is the set of all literals over Δ. Let

$\delta_1, \ldots \delta_n$ be literals, then $\delta_1 \wedge \ldots \wedge \delta_n$ are conjunctive terms. We call μ a *minterm* over a set of letters Δ, if μ is a conjunctive term over Δ and for each $\delta \in \Delta$ either δ or $\neg\delta$ appears in μ. $M(\Delta)$ is the set of all minterms over Δ. Let μ, μ' be conjunctive terms. We define the intersection of μ and μ' as the conjunction of all literals which appear both in μ and μ': $\mu \cap \mu' = \delta_1, \wedge \ldots \wedge \delta_n, \delta_i \in \{\delta \in \bar{\Delta}, \mu \rightarrow \delta, \mu' \rightarrow \delta\}$.

We define the size of a conjunctive term as the number of distinct literals it is composed of. Formally, let $D \subset V$ be the set of boolean decision variables. Then $size(\delta) = |\{d \in D | \delta \rightarrow d \vee \delta \rightarrow \neg d\}$ □

Definition 3 (*Labels*) Let $P = (N, E, V, A)$ be a process model, and $D \subset V$ be the set of boolean decision variables of P. $n.\lambda$ assigns a unique propositional letter $d \in D$ to each xor-split node $n \in N$. For each xor-split node x there are two outgoing edges e_1 and e_2 in E with $e_1.\delta = x.\lambda$ and $e_2.\delta = \neg x.\lambda$. All other edges have the label $true$.

The label $n.\Lambda$ of a node n is defined as

1. $n.\Lambda = \{true\}$, if n is a start node;
2. $n.\Lambda = x.\Lambda \wedge (x, n).\delta$, if $(x, n) \in E$ and x is an xor-split node;
3. $n.\Lambda = m_1.\Lambda \cap m_2.\Lambda$, if n is an xor-join node and $(m_1, n), (m_2, n) \in E$;
4. $n.\Lambda = m.\Lambda$, if $(m, n) \in E$ for any other node. □

Example 1 (*Example of labels*) In Fig. 3 a global process with labels for each node is shown. $A1_{P1}$ is unconditionally executed and has the label $true$. $A2_{P2}$ is conditionally executed if the $true$ branch of the xor-split is taken and therefore has the label $d1$. In analogy, $A3_{P3}$ has the label $\neg d1$. Finally, $A4_{P4}$ is unconditionally executed and has the label $true$. Note that the calculation of labels in Definition 3 (3) implies that the label of a node does not include propositional terms of previous xor-blocks. Accordingly, the label of $A4_{P4}$ is $true$. □

3.3 Augmented process model

Data exchange between different parties is implemented via communication steps. A process model with communication steps (a data-flow implementation) is called an augmented process model.

Definition 4 (*Augmented process model*) An *augmented process model* $M = (N, E, V, A)$ is a process model where nodes may also be of type s (communication step). For a communication step $s = (a, r, c, X)$ $a \in A$ represents the sender, $r \in A$ the receiver of a message which is sent under the condition c and transfers the variables $X \in \mathcal{P}(V)$, where c is a propositional term over the decision variables $D \subset V$ given in disjoint disjunctive normal form such that $c \rightarrow s.\Lambda$. □

During the generation of local process models for each participants, communication steps are either inserted as send steps in the local process of the sender and as receive steps in the local process of the receiver or they are realized as request steps in the local process of the receiver and as response steps in the local process of the sender. When an augmented process model with multiple actors is projected [13] to process models for each actor, additional message exchanges for handing over the control-flow are

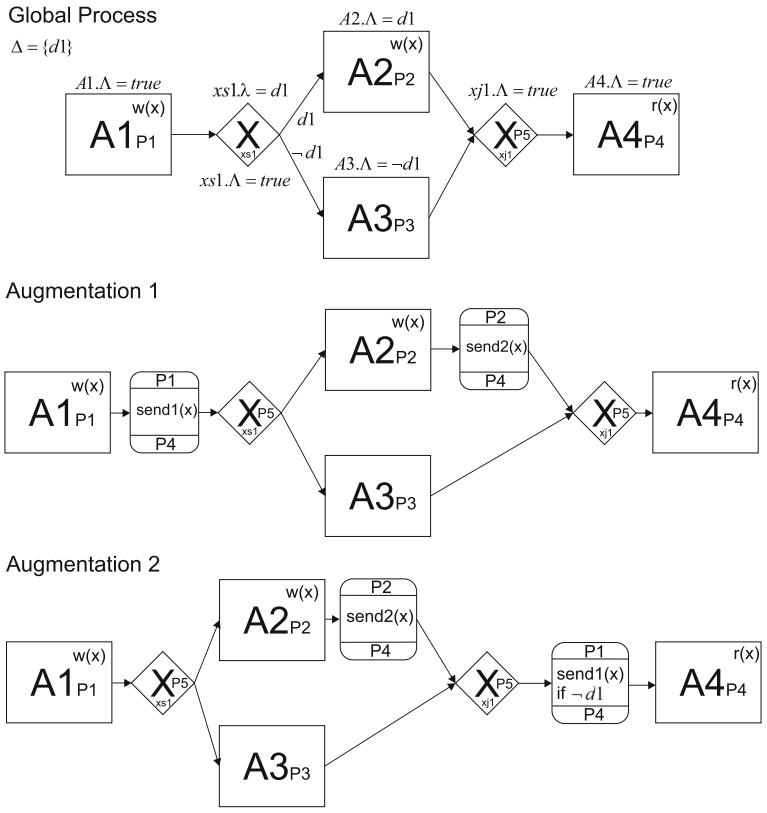

Fig. 3 Example global process with labels and two implementations (augmented processes)

added to the process model. This paper focuses on message exchanges for data-flow and leaves the partitioning of the control-flow to the partitioning phase [13].

Definition 5 (*Instance type*) Each minterm I over the set of decision variables D of a process P with start node s and end node e constitutes an *instance type* P^I which is defined as a connected subgraph of P which includes s and e where each xor-split node has exactly one successor and each xor-join has exactly one predecessor (depending on the value of the decision variable). □

Example In Fig. 3, we have two instance types constituted by the minterms $d1$ and $\neg\, d1$ over the decision variable $d1$: $A1_{P1}$, $A2_{P2}$, $A1_{P4}$ for $d1$ and $A1_{P1}$, $A2_{P3}$, $A1_{P4}$ for $\neg\, d1$.

The definition of an instance type does not directly correspond to valid runs or instances of a process as multiple instance types can specify the same run or process instance. The reason is that decision variables of non-executed gateways are not relevant since they have no influence on the selected branches of executed gateways.

In the augmented model data are transmitted from one actor to another by means of the communication steps. There is no global data store. For analyzing the correctness

of an augmentation, we consider the distributed execution of the process and the execution by only one actor. Informally, an augmentation is correct, if the data seen by each activity in both executions is the same.

The origin of a variable x of a node n in an instance type I is the closest predecessor of n in P^I which writes x. For every instance type, each input variable of each activity must have a single origin (task writing to that variable).

Definition 6 (*Origin*) The *origin* for an input variable x of a node n in an instance type P^I, denoted $o(P^I, n, x)$, is defined as a node m such that there exists a path p in P^I containing n and m and there is no step m' with x as output variable between n and m in p.

Formally, $o(P^I, n, x) = m$ iff $m \in n.pred_{P^I}+ \wedge \{m, n\} \subseteq P^I \wedge x \in m.w \wedge \forall m' \in n.pred_{P^I}+ : m' \in P^I \wedge x \in m.w \wedge m \in m'.succ_{P^I}+$. □

Definition 7 (*Correct process model*) A process model $P = (N, E, V, A)$ is correct, iff for each instantiation I of decision variables, for each input variable $x \in V$ of each activity node $n \in N$: $o(P^I, n, x)$ exists and is unique. □

This requirement covers the usual data-flow faults [25] like uninitialized variables and race conditions. In particular, we do not allow read-write or write-write dependencies between parallel tasks.

For the distributed execution of an augmented process we have to consider that a participant only can access the content of a variable if it was produced locally or if it was received through a communication step. The distributed origin thus traverses the graph along the communication steps to identify the last writer.

Definition 8 (*Distributed origin*) The *distributed origin* of the input variable x in node n of an instance type P^I, $o^d(P^I, n, x) = m$, is defined as follows: Let a be the actor or sender of n and let m be the closest predecessor step of n, where $n.type = activity\text{-}step$ or $(n.type = communication\text{-}step \wedge I \rightarrow n.c)$ with x as output parameter and a as actor (for activity steps) or recipient (for communication steps). If m is an activity step then $o^d(P^I, n, x) = m$, if m is a communication step $s = (b, a, c, X)$ then $o^d(P^I, n, x) = o^d(P^I, m, x)$. □

The recursive definition of the distributed origin allows to directly receive a variable from the origin or to receive a variable transitively via multiple communication steps. However, the distributed origin is always an activity step with x as output. We can now define the correctness of an augmentation.

Definition 9 (*Correct augmentation*) The augmentation P of a process is correct, iff for each instantiation I of decision variables, for each input variable x of each activity node $n \in N$: $o(P^I, n, x)$ exists and is unique and $o^d(P^I, n, x) = o(P^I, n, x)$. □

Example 2 (*Correct augmentation*) Augmentation 1 in Fig. 3 is correct. There are the following instance types: $A1_{P1}$, $send1 = (P1, P4, \{\}, \{x\})$, $A2_{P2}$, $send2 = (P1, P4, \{d1\}, \{x\})$, $A1_{P4}$ for the minterm $d1$ and $A1_{P1}$, $send1 = (P1, P4, \{\}, \{x\})$, $A2_{P3}$, $A1_{P4}$ for the minterm $\neg d1$. The distributed origin of x for $A4_{P4}$ for the minterm $d1$ is $A2_{P2}$, the distributed origin of x for $A4_{P4}$ for the minterm $\neg d1$ is $A1_{P1}$. This trivially equals the origin in the global process model. It is therefore a correct augmentation. □

4 Measures for augmentations

As motivated in Sect. 2.3 we assess augmentations based on the perspectives of additional message exchanges or interactions for implementing the data-flow, the overall amount of data transmissions, and the impact on the complexity of local processes. We have already introduced measures that provide indications for each perspective in [19]. In this section, we revisit and refine the measures α, β and γ for augmentations. In particular, we refine the measures based on the extended process model including labels newly introduced in this paper.

We have emphasized in Sect. 2.1 that it makes a difference whether some data transmission is executed in all cases or only in specific cases during runtime. Conditional communication steps will contribute less to the overall number of sent messages or transmitted variables at runtime. In order to provide a combined measure of conditional and non-conditional communication steps we introduce the weight function. It approximates the probability that a communication step is executed. Since we have no knowledge about branching probabilities, we assume that the probability of each outgoing branch of a xor-split is 50%.[2] For efficiently calculating the weight, we use the size of labels of communication steps.

Definition 10 (*Weight of a communication step*) Let sr be a communication step of some augmentation P. Let $C(sr.c)$ be the set of all conjunctive terms in $sr.c$.

$$weight(sr) = \sum_{c \in C(sr.c)} 2^{-size(c)}$$

□

Example 3 (*Weight of a communication step*) The weight of the communication step $send1$ of Augmentation 1 in Fig. 3 is $2^{-|\{\}|} = 1$. It is executed unconditionally. In contrast, the weight of $send2$ of the same augmentation is $2^{-|\{d1\}|} = 0.5$. It is executed in 50% of all instance types. In the same way, the weight of $send1$ in Augmentation 2 is $2^{-|\{d1\}|}$. $send1$ is not part of the xor-block but it has d in its condition. Therefore, its weight is also 0.5. □

4.1 Assessing the number of additional messages of a data-flow implementation

As shown in Sect. 2.3 one important criteria of an augmentation is the number of additional messages or interactions that are needed for realizing the cross-organizational data-flow. This property is assessed by the α measure.

In the later partitioning phase every edge (m, n) of the global process model, where the actor of m $(m.a)$ is not equal to the actor of n $(n.a)$ results in a message exchange between $m.a$ and $n.a$ for realizing the cross-organizational control-flow. When communication steps are placed between m and n in an augmentation of the global process, these steps can only be included in the control-flow message, if the sender is the actor

[2] Clearly, if additional knowledge about branching probabilities exists, a refined weight function could be used.

of m and the receiver is the actor of n. Otherwise additional interactions are required for each such communication step. In particular, if either the sender is the actor of m or the receiver is the actor of n, exactly one additional interaction is required for each such communication step (assuming send-receive style interactions if $m.a$ is the sender and request-response style interactions if $n.a$ is the receiver). This property holds for all classes of augmentations relevant for this paper. We now define when a communication step can be included in a control-flow message.

Definition 11 (*Includable communication step* $includable(c)$) Let c be a communication step of an augmentation. Let $p1$ be the actor of the closest non-communication step in $c.pred+$. Let $p2$ be the actor of the closest non-communication step in $c.succ+$. $includable(c) = c.a = p1 \wedge c.r = p2$

The $alpha$ measure is the weighted sum of all communication steps that will require additional interactions:

Definition 12 (*Measure* α) Let $P = (N, E, V, A)$ be an augmentation. Let $NI = \{n : n \in P.N \wedge \neg includable(c)\}$ be the set of all communication steps that cannot be included in the control-flow.

$$\alpha(P) = \sum_{s \in NI} weight(s)$$

Example 4 (*Measure* α) For Augmentation 1 in Fig. 2 we get an α value of 0.0 since the payload of all communication steps can be included in control-flow messages between participants. For Augmentation 2 in Fig. 2 we get an α value of $1+1+0.5 = 2.5$ since there are two unconditional communication steps and one conditional communication step which do not follow the control-flow. For Augmentation 4 in Fig. 2 we get an α value of $1 + 0.5 = 1.5$. □

4.2 Assessing the amount of data transmissions

Another important aspect of an augmentation is the total number of data transmissions. It provides an estimate on potential bandwidth requirements while still abstracting from actual sizes of data objects. For the β-measure all communication steps are of interest. It is defined as the weighted sum of all variable transmissions.

Definition 13 (*Measure* β) Let $P = (N, E, V, A)$ be an augmentation. Let S be the set of communication steps in $P.N$.

$$\beta(P) = \sum_{s \in S} weight(s) * |s.X|$$

□

Example 5 (*Measure* β) Augmentation 1 in Fig. 2 contains 5 unconditional communication steps with a total of 6 transmissions and one conditional communication step

transmitting one variable. The overall β value is 6+0.5=6.5. Regarding the Augmentation 2 in Fig. 2, there are 4 unconditional communication steps with a total of 4 variable transmissions and one conditional communication step sending one variable. The overall β value is 4+0.5=4.5. □

4.3 Assessing collaboration complexity

An augmentation has an impact on the number of decision variables required by each participant. We assume that the more decisions a partner needs only for correctly executing the data-flow, the more complex his local process will get. We use the γ value as an indicator for this additional complexity of local processes imposed by the augmentation. It is defined as the total number of decision variables needed by all actors for the augmentations and for the control-flow minus the total number of decision variables needed by all actors for the control-flow only. γ counts the overall number of additional decision variables for an augmentation of all actors.

Definition 14 (*Measure* γ) Let $P = (N, E, V, A)$ be an inter-organizational process, let $P' = (N, E, V, A)$ bet a correct augmentation of P.

Let $reqVar(Pr, a)$ be a function assigning the actor a of the process Pr to a set of propositional letters that a requires to participate in the process Pr. An actor requires a propositional letter $\delta \in \Delta$ if he is the actor, receiver or sender of at least one node n in $Pr.N$, where the label of n, $n.\lambda$ contains δ or if he is the actor or receiver of some communication step sr, where the simplified boolean formula of $sr.c$ contains δ.

$$\gamma(P) = \sum_{p' \in P'.A} |reqVar(p', P')| - \sum_{p \in P.A} |reqVar(p, P)|$$

□

Example 6 (*Measure* γ) For Augmentation 1 in Fig. 2 the γ-value is 0 since no partner needs an additional decision to execute the communication steps. For Augmentation 2 in Fig. 2, the γ value is 1 since the diagnosis institute gets part of the xor-block only for transmitting data to the rehabilitation specialist. □

5 Classes of augmentations

While there exist very large numbers of correct augmentations for inter-organizational processes, we are specifically interested in augmentations that obey certain properties. In Sect. 5.1 we revisit safeness-classes of augmentations based on the principle of information hiding. Another quality criteria for augmentations is the absence of obsolete transmissions. We define this class formally in Sect. 5.2.

5.1 Safeness classes of augmentations

For being self-contained we revisit safeness-classes for augmentations initially introduced in [19] here. In our motivating example in Fig. 2 some augmentations impose confidentiality issues. In Augmentation 1 the insurance company and the accounting office receive the MRT image only to forward it to other participants. In Augmentation 3, the rehabilitation specialist receives the MRT image even if the rehabilitation is not approved by the insurance. Both circumstances may impose non-acceptable confidentiality issues depending on user requirements. We now define different classes of augmentations that provide different levels of confidentiality.

Let s be a communication step where sender a sends a variable v with origin o to recipient r. We distinguish three spheres of variables: an actor is in the *static sphere* of the variable v, if he writes to or reads this variable anywhere in the process definition. An actor is in the *weak dynamic sphere* of v, if he reads the variable v with origin o in any step of the process reachable from s. An actor is in the *strong dynamic sphere*, if it is certain that in any continuation of the process he will read the variable v with origin o.

Definition 15 (*Static sphere*) Let $B' = (N, E, V, A)$ be an augmentation, let $sr = (a, r, c, X) \in B'.N$ be a communication step, let v be a variable in $sr.X$. The receiving actor r of sr is in the static sphere of v iff there exists a node $n \in B'.N$ where $n.type = activity$ and $n.a = r$ and $v \in n.v^r$. □.

Definition 16 (*Weak dynamic sphere*) Let $B' = (N, E, V, A)$ be an augmentation, let $sr = (a, r, c, X) \in B'.N$ be a communication step, let v be a variable in $sr.X$ with origin o. Let AI_{sr} bet the set of all instance types of B' containing sr. The receiving actor r of sr is in the weak dynamic sphere of v iff at least one instance type $P^I \in AI_{sr}$ contains some node $n \in B'.N$, where $n.type = activity$ and $n.a = r$ and $v \in n.v^r$ and $o(P^I, n, v) = o$ and n is a direct or transitive successor of sr. □

Definition 17 (*Strong dynamic sphere*) Let $B' = (N, E, V, A)$ be an augmentation, let $sr = (a, r, c, X) \in B'.N$ be a communication step, let v be a variable in $sr.X$ with origin o. Let AI_{sr} bet the set of all instance types of B' containing sr. The receiving actor r of sr is in the strong dynamic sphere of v iff every instance type $P^I \in AI_{sr}$ contains some node $n \in B'.N$, where $n.type = activity$ and $n.a = r$ and $v \in n.v^r$ and $o(P^I, n, v) = o$ and n is a direct or transitive successor of sr. □

We use these definitions of spheres to define classes of augmentations:

Definition 18 (*Statically safe augmentation*) An augmentation is statically safe, iff for every communication step, the receiver is in the static sphere of the transmitted variable. □

Definition 19 (*Weak dynamically safe augmentation*) An augmentation is weak dynamically safe, iff for every communication step, the receiver is in the weak dynamic sphere of the transmitted variable. □

Definition 20 (*Strong dynamically safe augmentation*) An augmentation is strong dynamically safe, iff for every communication step, the receiver is in the strong dynamic sphere of the transmitted variable. □

Example 7 (*Safeness classes*) Augmentation 1 in Fig. 2 is not statically safe. The accounting office receives the MRT image but it does not execute any activity reading it. Augmentation 3 in Fig. 2 is weak dynamically safe but it is not strong dynamically safe because the rehabilitation specialist receives the MRT image even if the rehabilitation is not approved by the insurance. Augmentation 2 and 4 in Fig. 2 are strong-dynamically safe. Only participants that certainly require some data value get access to it. □

5.2 Non-redundant augmentation

As already discussed in in Sect. 2.3 augmentations should not contain message exchanges that do not contribute to implement the cross-organizational data-flow. We now define the class of non-redundant augmentations formally. We first define a canonical form for augmentations.

Definition 21 (*Augmentation in canonical form*) An augmentation P is in canonical form, if every communication step sr in $P.N$ has the following form: $|sr.X| < 2$ and $sr.c$ is not a disjunction of terms.

Every augmentation can be transformed to an augmentation in canonical form by splitting every communication step with more than one variable to separate communication steps for each variable (equivalence transformation rule $TS5$ in [18]) and splitting every communication step into one for each member of the disjunctions of its condition ($TS6$ in [18]).

Definition 22 (*Redundant transmission*) Let P be an augmentation and $P' = (N, E, V, A)$ be the canonical form of P. Let $sr = (a, r, c, X)$ be a communication step in $P'.N$. sr is redundant, iff removing sr from P' still leads to a correct augmentation. □

Definition 23 (*Non-redundant augmentation*) An augmentation P' is non-redundant, iff there exists no redundant transmission in P'. □

From Definitions 22 and 9 follows that there are two cases of redundant transmissions: redundancies due to the absence of reading activities and redundancies due to double transmissions of the same data values.

No-reader redundancies occur, if a value transmitted by a communication step is never read by some succeeding activity-step.

Definition 24 (*No-reader redundancy*) Let P be an augmentation and $P' = (N, E, V, A)$ be the canonical form of P. Let sr be a communication step in $P'.N$. Let PI be the set of all instance types of P' containing sr such that $sr.c$ evaluates to true. sr causes a no-reader redundancy, if there exists no instance type P^I in PI containing some activity step r in $sr.succ+$ and $o^d(P^I, sr, v) = o^d(P^I, r, v)$ and sr directly or indirectly transmits v to r in P^I. □

In order to define double transmission redundancies, we first define, when an actor at some step is in the possession of some variable:

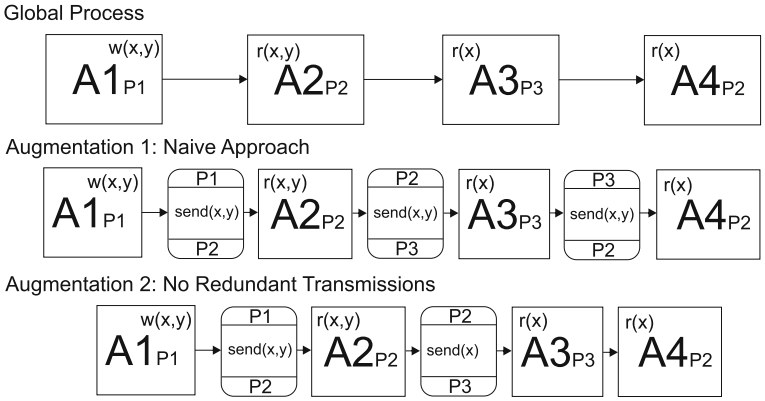

Fig. 4 Redundant versus non-redundant augmentation of a process

Definition 25 (*Actor in possession of a variable at step*) An actor p is in possession of some variable v under condition c at some step r, where p is an actor or receiver of r, iff for every instance type P^I containing r that is permissible by c: $o^d(P^I, r, v) = o(P^I, r, v)$ holds. The predicate $inPos(p, v, c, r)$ returns true iff p is in possession of v under condition c. □

Definition 26 (*Double-transmission redundancy*) Let P be an augmentation and $P' = (N, E, V, A)$ be the canonical form of P. A communication step $sr \in P'.N$ with $sr.X = \{v\}$ is redundant, iff for every instance type P^I containing sr such that $I \rightarrow sr.c$ and $o(P^I, sr, v) = m$, there exists a node n, where $n \in m.succ+ \wedge n \in sr.pred+$ and n has $sr.r$ as actor or receiver and $sr.r$ is in possession of v at step n under condition $sr.c$. □

Example 8 (*Redundant transmissions*) Fig. 4 shows a global process with 4 activities executed by 3 different actors. First, variable x and y are updated by $A1_{P1}$. Then $A2_{P2}$ reads both variables. Later $A3_{P3}$ and $A4_{P2}$ read x. In the naive implementation (Augmentation 1), every variable is transmitted on every inter-actor control-flow edge. The solution contains multiple redundant transmissions: sending y from $P2$ to $P3$ causes a no-reader redundancy (see Definition 24). Sending x (and y) from $P3$ to $P2$ is redundant according to Definition 26 since $P2$ is already in the possession of x. Augmentation 2 is an alternative, non-redundant augmentation for the same input process. □

Note that double transmission redundancies can occur on a single path between origin and reader in a process model like in Fig. 4 but they can also occur due to communication steps of different paths, if the process model contains parallel gateways. An example is shown in Fig. 6. If the variable x is transferred from $A1$ to $A5$ by inserting communication steps in path 3, then an additional augmentation of any other path leads to double transmission redundancies.

6 A heuristic approach for generating augmentations

Existing base-line strategies for augmentations optimize only one of the measures but have a negative impact on other measures [19]. The data-flow follows control-flow strategy produces augmentations with an α measure of 0. However, such augmentations have the penalty of very high β measures and they cannot generally obey the safeness classes. The direct-send strategies $B2$ and $B3$ produce augmentations with low β measures but these augmentations suffer from high α and γ measures. It is therefore beneficial to apply a flexible strategy. This may lead to augmentations such as augmentation 4 in Fig. 2.

We have therefore designed a heuristic approach which is guided by the measures. It was designed with the following objectives:

- Generation of non-redundant augmentations (see Sect. 5.2).
- Support for all safeness classes (see Sect. 5.1).
- Generation of solutions that optimize the α and β measures while limiting negative effects on the γ measure.
- Feasible performance for large classes of real world inter-organizational processes.

The requirement for feasibility for large classes of real-world inter-organizational processes does not allow to perform an exhaustive search over the solution space defined by [18]. Therefore, we have developed a heuristic approach that aims at an optimized integration of the data-flow into the control-flow to reduce the α and β values. If such a solution is not possible due to the required safeness class for a specific transfer, a variant of late-send that improves the γ value is applied.

The approach generates augmentations in two steps. First a pre-solution in canonical form is generated by incrementally augmenting the required paths between origins and their readers. In a second step further optimizations using a subset of the rules from [18] are applied. We first informally discuss optimizations of the data-flow follows control-flow approach in Sect. 6.1. We then introduce the required formalization of our approach and proof that the incremental application of augmentations leads to correct and non-redundant augmentations in Sect. 6.2. Finally, we present the whole approach in Sect. 6.3.

6.1 Optimizing data-flow follows control-flow

A straightforward implementation of the data-flow follows the control-flow base-line approach is correct but not optimal with regard to the α and β measures due to two sources of redundancies: (a) No-reader redundancies (see Definition 24), and (b) Double-transmission redundancies (see Definition 26).

6.1.1 Optimizations on single paths

In a purely sequential process, a straightforward implementation of the data-flow follows control-flow approach results in redundant transmissions, if one actor is assigned to at least two steps that are not directly preceding between some origin o and some

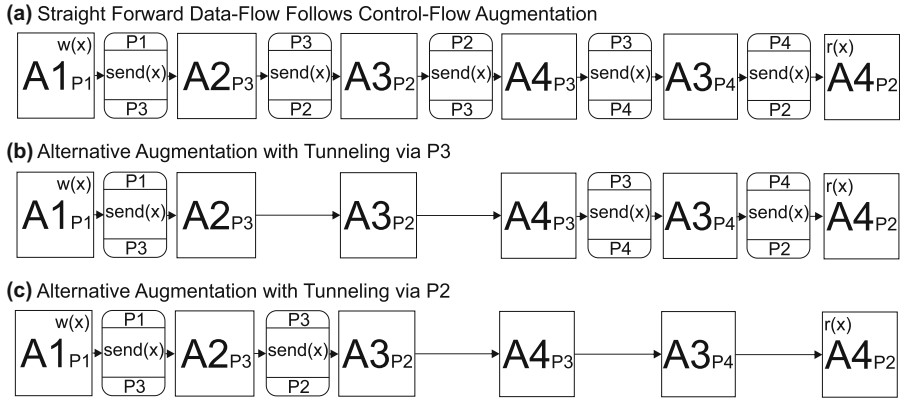

Fig. 5 **a** Straightforward data-flow follows control-flow, **b** tunnel via P3 **c** tunnel via P2

reader r. We refer to this effect as the existence of *tunnels* between different steps of the same actor. On the one hand redundancies should be avoided and on the other hand tunnels provide room for optimizations.

Example 9 (*Tunneling*) Fig. 5 shows three augmentation of a sequential process. The first step $A1_{P1}$ writes to variable x and only the last step $A4_{P2}$ reads x. In the straightforward implementation (a), x is forwarded via every control-flow edge. Therefore, actors $P3$ and $P2$ receive the same variable (values) twice. Alternative augmentations are (b) requiring 3 communication steps ($\alpha' = 3$, $\beta = 3$) and exploiting the tunnel of $P3$ and augmentation (c) requiring only 2 communication steps ($\alpha = 2$, $\beta = 2$) exploiting the tunnel of $P2$. It is important to note that augmentations (b) and (c) are both non-redundant. □

Obviously, an advanced data-flow follows control-flow approach should make use of *tunnels* to optimize the β value. Tunnels do not only occur on completely sequential processes. In case of xor-gateways, multiple different paths exist where each path may contain tunnels. In the absence of parallel gateways, clearly each paths between origin and reader needs to be augmented to obtain a correct augmentation.

6.1.2 Influence of parallel gateways

For processes containing parallel gateways, additional room for improvements but also additional restrictions on the paths that may be augmented exist. Parallel gateways may lead to additional redundancies of straightforward data-flow follows control-flow implementations. The general problem is that if the data-flow is routed over one branch of par-split and par-join nodes, then an additional routing over the other branch is redundant.

Example 10 (*Influence of parallel gateways*) Figure 6 shows an example, where an augmentation is required between the origin $A1_{P1}$ and the reader $A5_{P5}$. There are the following paths between $A1_{P1}$ and $A5_{P5}$: *p1=(A1, ps1, xs1, A2, xj1, pj1, A5)*,

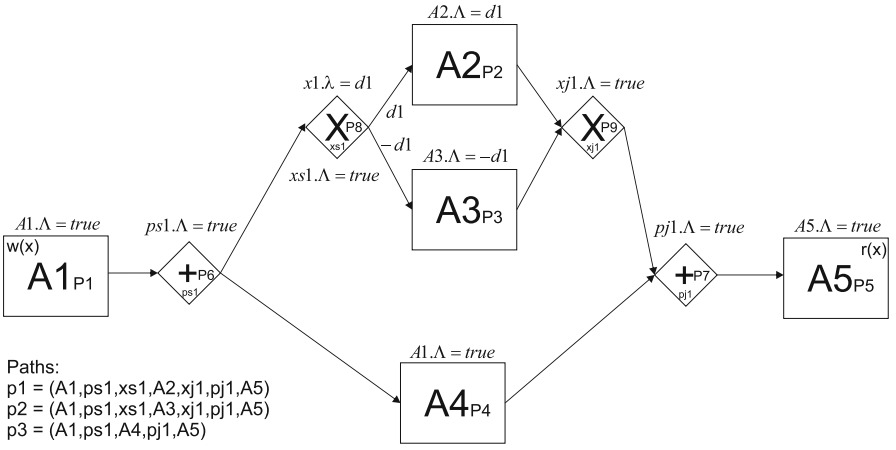

Fig. 6 Example of a process model with 2 augmentation candidates

p2=(A1, ps1, xs1, A3, xj1, pj1, A5), *(p3=A1, ps1, A4, pj1, A5)*. A straightforward implementation forwards the data over each path. However, if the data is routed via $p3$, then an additional routing via $p1$ or $p2$ is redundant since $A5_{P5}$ is already in the possession of the data of $A1_{P1}$. When routing via $p1$ and $p2$, then an additional routing via $p3$ is redundant. □

The correctness criteria for data-flow requires that the origin of some variable and some reader in some instance type is always unique. This has also consequences on the computation of augmentations. We present the phenomena based on an example. We now assume that $A2_{P2}$ in Fig. 6 also writes to x. Since the origin must be unique for each instance type for the reader $A5_{P5}$, the path ($A1_{P1}$, $ps1_{P6}$, $A4_{P4}$, $pj1_{P7}$, $A5_{P5}$) must not be used to forward x from $P1$ to $P5$ since the value of x is potentially updated in the other branch of $ps1$.

An augmentation procedure must only augment paths which do not lead to non-unique origins and as discussed earlier, it should avoid double transmission redundancies due to parallel gateways. Finally, it should use potential tunnels for optimization.

6.2 Formalization

We now introduce the necessary formalization of our approach and then prove the correctness and non-redundance of the generated augmentations. Our algorithm is based on incrementally adding augmentations for all required paths between origins and readers. We refer to such paths as $Augmentation\ Paths$:

Definition 27 (*Augmentation path*) Given an augmentation P' for a process P (including $P' = P$).

An augmentation path $path(P', o, r, v, c)$ is a path in form of a subgraph of the Augmentation P' for some variable $v \in P'.V$, an activity $o \in P'.N$ writing to v

($v \in o.w$), and an activity $r \in P'.N$ reading v ($v \in r.r$) where there is no other activity writing to v in-between and $r.a \neq o.a$. The condition c of the path p' is the conjunction of all labels of all non-communication-steps in the path.

Formally, $path(P', o, r, v, c)$ is a connected subgraph p' of P' such that:

- $o.pred = \emptyset \wedge r.succ = \emptyset \wedge \forall n \in p'.N \backslash \{o, r\} : |n.succ| = 1 \wedge |n.pred| = 1$;
- $\forall n \in p'.N \backslash \{o, r\} : n.type = activity \Rightarrow v \notin n.w$;
- $c = \bigwedge \{n.\Lambda : n \in p'.N \wedge n.type \neq communication - step\}$. □

We have already informally discussed the influence of parallel gateways on the paths which may be augmented to obtain a correct augmentation of the process in Sect. 6.1.2. We now define the set *Augmentation Candidates* as the set of all augmentation paths that can be augmented to obtain a correct augmentation of a process. Basically, an augmentation path $p1$ with origin $o1$ is in the set of augmentation candidates if there is no other augmentation path $p2$ with origin $o2$ and the same reader and variable where $o2$ is a predecessor of $o1$ and the process model permits a parallel instantiation of both paths.

Definition 28 (*Augmentation candidates*) The set of all augmentation candidates AC_P of a process P with correct data-flow is defined as follows:

$$\begin{aligned} AC_P = \{p1 : p1 = path(P, o1, r, v, c) \wedge \nexists p2 \\ = path(P, o2, r, v, c2) : o2 \in o1.succ_P + \\ \wedge o1 \neq o2 \wedge (p1.c \rightarrow p2.c \vee p2.c \rightarrow p1.c)\} \end{aligned}$$

□

The solution space for the augmentations following the control-flow of some augmentation path is expressed by the *Augmentation Graph*.

Definition 29 (*Augmentation graph AUG*) Given a process P, an Augmentation P' of P and an augmentation path $p = path(P', o, r, v, c)$.

The augmentation graph $AUG = (N, E, o, r, p, v)$ is a directed weighted graph, where the set of nodes is $N \subseteq \{n : n \in p.N \wedge n.type \neq communication\text{-}step\}$. E is the set of weighted edges. An edge (a, b, w) is in E, iff one of the following conditions holds:

1. *b is in possession of v:* $a = o \wedge inPos(b.a, v, c, b)$. The weight of the edge, w is 0.0.
2. *There is a tunnel between a and b:* $a.a = b.a \wedge a \in b.pred_p+$. The weight of the edge, w is 0.0.
3. *A communication step between a and b is allowed:* Let c' be the condition of a potential communication step between a and b defined as $c' = \bigwedge \{t.\Lambda : t \in (o.succ+ \cap b.pred+) \cup \{o, b\} \wedge t.type \neq communication\text{-}step\}$. Let $X = \{x : x \in b.succ+ \wedge x.a = b.a \wedge \nexists w \in (x.pred+ \cap b.succ+) : v \in w.w\}$. A communication step between a and b is allowed, iff all of the following conditions hold:

(1) a and b are assigned to different actors ($a.a \neq b.a$);
(2) a is a direct predecessor of b in P and sending v from a to b under condition c' complies with the given safeness class;
(3) $\nexists\, x \in X : inPos(x.a, v, c' \wedge x.\Lambda, x)$.

Iff all 3 conditions hold, (a, b, w) is in E, where w is the weight of a potential communication step between a and b in P with condition c' (given in Disjoint Disjunctive Normal Form). □

Every path from o to r with a minimal number of edges with weight $w > 0$ in AUG determines a transitive augmentation solution.

Definition 30 (*Transitive augmentation solution*) Given an augmentation graph $AUG = (N, E, o, r, p', v)$ for an augmentation path $p' = path(P', o, r, v, c)$. Let p_{sol} be a path from o to r in AUG in form of a connected subgraph of AUG where the number of edges with weight $w > 0$ is minimal.

- If p_{sol} does not exist, then there exists no transitive augmentation solution for p' in P'.
- Otherwise, the augmentation solution sol for p' is derived by adding communication steps to sol for every edge in $p_{sol}.E$ where the weight is > 0. The condition of each added communication-step is the conjunction of the step labels of all previous non-communication-steps in p'.
  Formally, sol is derived from p' by applying the following rule:

$$\forall (m, n, w) \in p_{sol}.E : w > 0 \Rightarrow \exists s \in sol.N :$$
$$s.pred_{sol} = \{m\} \wedge s.succ_{sol} = \{n\} \wedge m.succ_{sol} = \{s\} \wedge n.pred_{sol} = \{s\}$$
$$\wedge s.type = communication - step \wedge s.X = \{v\} \wedge s.a = m.a \wedge s.r = n.a \wedge$$
$$s.c = \bigwedge \{t.\Lambda : t \in s.pred_{p'} + \cup \{s\} \wedge t.type \neq communication\text{-}step\}$$

□

Clearly, an augmentation solution can be equivalent to the augmentation path if r is already in the possession of v. Transitive augmentation solutions represent optimized variants of (base-line B1).

If no transitive solution exists for some path, we apply a direct augmentation solution as a variant of late-send (base-line B2) for this path. However, in contrast to $B2$, any actor of a step who is already in possession of the variable can act as a sender.

Definition 31 (*Direct augmentation solution*) Given an augmentation path $p' = path(P', o, r, v, c)$. Let O be the set of all actors who can act as senders of the variable: $O = \{s.a : s \in p'.N \wedge s.type \neq communication\text{-}step \wedge inPos(s.a, v, c, s)\}$.

- If $r.a \in O$ then the augmentation solution $sol = p'$.
- Otherwise: let $se \in O$; let cs be some node where $cs.type = communication\text{-}step \wedge cs.a = se \wedge cs.r = r.a \wedge cs.X = \{v\} \wedge cs.c = c$.
  The solution sol is derived from p' by applying the following rule: $\exists m \in r.pred \Rightarrow m.succ_{sol} = \{cs\} \wedge r.pred_{sol} = \{cs\}$. □

Note that a direct augmentation solution always exists since the origin is trivially in possession of the variable.

Definition 32 (*Augmentation solution*) Let $sol(p, P', v)$ be a function returning either one transitive or one direct augmentation solution for the augmentation path p of P', and variable v. □

Definition 33 (*Application of an augmentation solution*) Given an augmentation G', an augmentation path p' of G' and an augmentation solution sol for p'. The application of sol to G', denoted as (G', sol) leads to a new augmentation G'' that contains all the added communication steps of sol.

Formally, $(G', sol) \rightsquigarrow G'' : G''.N = G'.N \cup sol.N \wedge G''.E = G'.E \setminus p'.E \cup sol.E$. □

Definition 34 (*Correct augmentation of a path*) Given a process P, an augmentation P' of P and an augmentation path $p' = path(P', o, r, v, c)$.

The path p' is correctly augmented in P' for variable v ($correct(p', P', v)$) if for all instantiations of decision variables $I_1, \ldots I_n$ of P' such that $I_i \rightarrow c$ the following holds: $o(P^{I_i}, r, v) = o^d(P^{I_i}, r, v) = w$. □

Definition 35 (*Projection of an augmentation path*) Given an augmentation G' and augmentation path p in G'.

The projection $\pi(p)$ of p is an augmentation path p' containing no communication steps such that all nodes in p' have the same topological order as in p. □

Obviously, let P be a process model, P' be an augmentation of P, p be an augmentation path in P', then $\pi(p)$ is a subgraph of P.

We now postulate that the application of any augmentation solution for some augmentation path p in an initial augmentation (with no communication steps inserted yet) will result in the correct augmentation of the path in the derived augmentation.

Theorem 1 (Correctness of initial augmentation solution) *Let P be a process; let AC_P be the set of augmentation candidates of P. Let $v \in P.V$; let $p = path(P, o, r, v, c) \in AC_P$; let $sol = sol(p, P, v)$.*

$$(P, sol) \rightsquigarrow P' \Rightarrow correct(p, P', v)$$

Proof 1 Let $S = sol.N \backslash p.N$; let AUG be the augmentation graph for P and p.

Case 1 sol is a direct augmentation solution.
By Definition 31, $|S| = 1 \wedge \exists s \in S : c \rightarrow s.c \wedge s.a = o$. Therefore, for every instantiation $P^I o^d(P^I, r, v) = o(P^I, r, v) = o \Rightarrow correct(p, P', v)$.

Case 2 sol is a transitive augmentation solution.
We prove by contradiction. Suppose $(P, sol) \rightsquigarrow P' \Rightarrow \neg correct(p, P', v)$. Therefore, there exists no path sg from o to r in form of a connected sub-graph of AUG such that $\forall (n, m, w) \in sg.E$ with $w > 0 : \exists s \in S : (n, s) \in sol.E \wedge (s, m) \in sol.E \wedge c \rightarrow sg.c$. This contradicts with Definition 30. □

We now postulate that the application of any further augmentation solution for some augmentation path leads to a correct augmentation of that path and renders no previously existing solution incorrect.

Theorem 2 (Non-interfering augmentations) *Let* $p1$ *and* $p2$ *be any augmentation paths of a process and* $sol1$ *and* $sol2$ *be the corresponding augmentation solutions. If applying* $sol1$ *and* $sol2$ *in isolation leads to correct augmentations of* $p1$ *and* $p2$*, also any incremental application will lead to a correct augmentation of* $p1$ *and* $p2$*. Formally, given a Process* P*, an Augmentation* P' *of* P*, a variable* $v \in P.V$ *and the set of all augmentation candidates* AC_P*.*
Let $\{p1, p2\} \subset AC_P : p1 = path(P', o1, r1, v, c1) \wedge p2 = path(P', o2, r2, v, c2)$*; let* $sol1 = sol(p1, P', v)$*; let* $sol2 = sol(p2, P', v)$*.*
$(P', sol1) \rightsquigarrow P1' \wedge correct(sol1, P1', v) \wedge (P', sol2) \rightsquigarrow P2' \wedge correct(sol2, P2', v)$
$\Rightarrow$
Let $p21 = path(P1', o2, r2, v, c2) : \pi(p21) = \pi(sol2)$*; let* $sol21 = sol(p21, P1', v)$*.*
$(P1', sol21) \rightsquigarrow P1'' \wedge correct(sol21, P1'', v) \wedge \exists p11 = path(P1'', o1, r1, v, c1) : \pi(p11) = \pi(sol1) \wedge \exists sol11 = sol(p11, P1'', v) \wedge correct(sol11, P1'', v)..$ □

Proof 2 We have to consider the following cases:

1. $o1 = o2$.
   We prove by contradiction. Let $r \in \{r1, r2\}$. Assume that there exists some $I : o^d(P1''^I, r, v) \neq o(P1''^I, r, v)$. From Definition 33 follows $\forall s \in sol1.N : \exists s \in sol21.N$ (no steps are removed). From $correct(sol1, P1', v)$ follows $\exists s \in sol21.N \backslash P1'.N : o^d(P1''^I, s, v) \neq o1$.

   Case 1.1 $sol21$ is a transitive augmentation solution. Let AUG be the augmentation graph for $P1'$ and $p21$; let n be the closest predecessor of s in $P1''$ where $n.type \neq communication\text{-}step$; let m be the closest successor of s in $P1''$ where $n.type \neq communication\text{-}step$. From Definition 30 follows: $\exists (n, m, w) \in AUG.E : w > 0$. This contradicts with Definition 29.

   Case 1.2 $sol21$ is a direct augmentation solution. $o^d(P1''^I, r, v) \neq o1$ directly contradicts with Definition 31.
2. $o1 \neq o2 \wedge r1 = r2$.

   We prove by contradiction. Let $N1, N2$ be the sets of communication steps for $sol1$, resp. $sol21 : N1 = sol1 \backslash \pi(sol1)$; $N2 = sol21 \backslash \pi(sol21) \backslash sol1$.
   For an interference between $sol1$ and $sol21$ there must be a communication step transmitting v which is executed when $o1$ is the origin and if $o2$ is the origin.

$$\exists s \in N1 \cup N2 : v \in s.X \wedge (o1.\Lambda \rightarrow s.c) \wedge (o2.\Lambda \rightarrow s.c)$$

   Case 2.1 $o1.\Lambda \nrightarrow o2.\Lambda \wedge o2.\Lambda \nrightarrow o1.\Lambda$.

   From Definitions 30, 31 directly follows: $\forall (n1, n2) \in N1 \times N2 : v \in n1.X \wedge v \in n2.X \Rightarrow n1.c \wedge n2.c = false$. Therefore, s cannot exist.

Case 2.2 $o1.\Lambda \rightarrow o2.\Lambda \vee o2.\Lambda \rightarrow o1.\Lambda$.

$\exists(n1, n2) \in N1 \times N2 : v \in n1.X \wedge v \in n2.X \wedge (n1.c \rightarrow n2.c \vee n2.c \rightarrow n1.c)$. From Definitions 30, 31, 28 follows $\pi(sol21) \notin AC_P \vee \pi(sol1) \notin AC_P$. Therefore, s cannot exist.

3. $o1 \neq o2 \wedge r1 \neq r2$.

Case 3.1 $p1.N \cap p2.N = \emptyset$. From Definition 33 follows that $sol21 = sol2$ and $sol2$ is a subgraph of $P2'$ and $sol1$ is a subgraph of $P1''$.

$$correct(sol1, P1', v) \wedge correct(sol2, P2', v) \Rightarrow$$
$$correct(sol1, P1'', v) \wedge correct(sol2, P1'', v)$$

Case 3.2 $p1.N \cap p2.N \neq \emptyset$.

Case 3.2a $o1.\Lambda \nrightarrow o2.\Lambda \wedge o2.\Lambda \nrightarrow o1.\Lambda$: see Case 2.1

Case 3.2b $o1.\Lambda \rightarrow o2.\Lambda \vee o2.\Lambda \rightarrow o1.\Lambda$: see Case 2.2 □

We have shown that the incremental application of augmentation solutions to two paths leads to correct augmentations for these paths without invalidating the previous solution. We now postulate that the incremental application of augmentation solutions of every augmentation candidate in a process leads to a correct augmentation for the process.

Theorem 3 (Correct incremental augmentation of a process) *Let P be a process model; let AC_P be the set of augmentation candidates of P.*
Let $p_1 = path(P, o_1, r_1, v_1, c_1), \ldots, p_n = path(P, o_n, r_n, v_n, c_n)$ be all augmentation paths in AC_P. $(P, sol(p_1, P, v_1)) \rightsquigarrow P'_1, \ldots, (P_{n-1}, sol(p_n, P'_{n-1}, v_n)) \rightsquigarrow P'_n$
Then P'_n is a correct augmentation of P. □

Proof 3 P'_1 is correct (see Theorem 1). Subsequent incremental applications of augmentation solutions do not interfere (see Theorem 2). The set AC_P covers all paths for all readers, origins and variables as required by Definition 9. □

We now postulate that the incremental application of augmentation solutions in a topological order of the readers to a process model results in a non-redundant augmentation of that process model.

Theorem 4 (Non-redundant augmentation solution) *Let P be a process model; let AC_P be the set of augmentation candidates of P.*
Let $p_1 = path(P, o_1, r_1, v_1, c_1), \ldots, p_n = path(P, o_n, r_n, v_n, c_n)$ be all augmentation paths in AC_P, topologically ordered by r_i in P.

$$(P, sol(p_1, P, v1)) \rightsquigarrow P'_1, \ldots, (P_{n-1}, sol(p_n, P'_{n-1}, vn)) \rightsquigarrow P'_n.$$

Then P'_n is a non-redundant augmentation of P. □

Proof 4 We prove by contradiction.
We assume there exists some solution $redsol = sol(p_m, P_{m-1}, v_m)$ for some augmentation path $p_m = path(P'_{m-1}, o_m, r_m, v_m, c_m)$ introducing redundancies. Let NS be the set of newly added communication steps in $redsol$: $NS = redsol.N \backslash P'_{m-1}.N$.
Case 1 Some newly introduced step ns is redundant.

$$\exists ns \in NS \wedge \exists pred \in ns.pred_{redsol}+ : pred.a = ns.r \wedge inPos(pred.a, v_m, c_m, pred)$$

Case 1.1 redsol is a transitive solution
Let AUG_m be the augmentation graph of p_m and v_m for augmentation P'_{m-1}; let p_{AUG} be the path from o to r in AUG_m that has led to $redsol$; let la be the closest non-communication-step in $ns.pred_{redsol}+$; let ra be the closest non-communication-step in $ns.succ_{redsol}+$. From $ns \in P'_m.N$ follows: $\exists p2_{AUG}$ as a subgraph of AUG_m such that: $(\forall (a,b,w) \in p_{AUG}.E : a \neq la \wedge b \neq ra : \exists (a,b,w) \in p2_{AUG}.E) \wedge (\exists (la, ra, 0) \in p2_{AUG}.E) \wedge (\forall (a,b,w) \in p2_{AUG}.E : a \neq la \wedge b \neq ra : \exists (a,b,w) \in p_{AUG}.E)$
$|\{(a,b,w) \in p2_{AUG}.E : w > 0\}| < |\{(a,b,w) \in p_{AUG}.E : w > 0\}|$
By Definition 30 $redsol$ cannot exist since $p2_{AUG}$ has less edges with weight > 0 than p_{AUG} and an augmentation solution is defined by a shortest path in the augmentation graph.

Case 1.2 redsol is a direct solution
If ns is redundant, then $inPos(r_m.a, v_m, c_m, r_m)$ holds in P'_{m-1}. By Definition 31 $redsol$ cannot exist.

Case 2.1 Some step $s \in NS$ renders a step $rs \in s.succ+$ redundant and $redsol$ is a transitive solution. Let $X = \{x : x \in s.succ + \wedge x.a = s.r \wedge \nexists w \in x.pred+ \cap s.succ+ : v \in w.w\}$. Therefore $rs \in X \wedge inPos(rs.a, m_v, s.c \wedge rs.\Lambda, rs)$ holds in P'_{m-1}. This contradicts with Requirement 3 of Definition 29. Thus, $redsol$ cannot exist.

Case 2.2 Some step $s \in NS$ renders a step $rs \in s.succ+$ redundant and $redsol$ is a direct solution. $rs.a = r_m.a \wedge rs \in r_m.succ$. Since augmentation is realized in a topological order $inPos(r_m.a, v_m, c_m, r_m)$ also holds in P'_{m-1}. Therefore, $redsol$ cannot exist.

6.3 Algorithm

We have shown that an incremental augmentation of all augmentation candidates of a process leads to a correct and non-redundant augmentation of the process. We have designed an algorithm that is based on incrementally calculating and applying transitive and direct augmentation solutions for each augmentation path. Since there may exist multiple transitive and multiple direct solutions for each augmentation path and there is some freedom in selecting the paths for augmentation (as discussed in Sect. 6.1.2), additional optimizations are possible.

For finding an optimal transitive solution for one single augmentation path, a solution with a minimal β value should be chosen. However, when incrementally computing augmentations for all paths between some origin and some reader, augmentations of single augmentation paths are not independent from each other. Therefore, the order in which the different paths are augmented has an impact on the overall result. Clearly, obtaining an optimized overall solution for a combination of one origin and one reader could be realized by computing augmentations for every order of paths between the origin and the reader. Since the number of paths grows exponentially with the number of gateways between origin and reader, such an approach is not feasible.

Our aim is to develop a feasible algorithm. Therefore, we apply heuristics. For augmenting single augmentation paths, we select a transitive solution (with $\alpha = 0$) that has a minimal γ value. By definition, transitive solutions contain a minimal possible number of communication steps. If a solution with $\alpha = 0$ does not exist, we apply a direct augmentation solution with a minimal γ value.

By implementing solutions with minimal numbers of communication steps and a minimum γ value we guarantee non-redundant solutions and we try to avoid solutions where additional actors need to know from control-flow decisions. This also implies that we favor solutions with more general conditions on communication-steps. Therefore, more steps are in the possession of variables. This is beneficial for the augmentation of future paths since it allows for reuse.

For computing a solution for all paths between some origin and some reader, we implement paths first that potentially have an $\alpha = 0$ score and potentially require a minimal number of communication steps. If multiple such solutions exist, the one with the minimal γ score is implemented first.

For tuning purposes, we exclude paths from the augmentation which are in conflict with already inserted ones. This allows to reduce the number of required augmentations without always computing the augmentation graph. Basically, if there is an augmentation for one branch of a parallel block, then an augmentation for a path that includes the other branch is not required as the reader is already in the possession of the variable.

Definition 36 (*Conflicting path*) Let P be the set of all augmentation paths in a process graph between the origin o and the reader r. The path $p1 \in P$ is in conflict with the path $p2 \in P$ iff:

$$\exists s1 \in p1.N, \exists s2 \in p2.N : s1.type = and - join \wedge s1 = s2 \wedge s1.pred \neq s2.pred$$

After the general principles of the augmentation method are introduced we can now provide the algorithm. For augmenting a process, we incrementally augment single augmentation paths between origins and readers. The computation of an augmentation solution for a single path is presented in Sect. 6.3.1. The augmentation of a process is discussed in Sect. 6.3.2.

6.3.1 Computing an augmentation solution for a single path

We illustrate the basic approach of finding an augmentation solution of a single augmentation path. We discuss a non-optimized variant to emphasize on the essentials of the approach. It calculates a transitive or direct augmentation solution adhering to the safeness class $mode$ for a single path p in a process model P between the origin o and the reader r for variable v. The safeness-class may be $none$ for no restriction, ssa for statically safe (see Definition 18), wds for weak dynamically safe (see Definition 19) and sds for strong dynamically safe (see Definition 20).

In a first step, the algorithm checks if r is already in the possession of v with origin o for the current condition c using $checkPosession(o, r, v, c, P, inPos)$ (see Definition 25). If yes, no augmentation is required and the path p is returned as an empty solution. In our implementation $checkPosession(o, r, v, c, P, inPos)$ is efficiently implemented by querying a lookup table $inPos(o, v, sr)$. The table is incrementally populated during the insertion of communication steps (see Sect. 6.3.2 for details).

If r is not in the possession of v, an augmentation solution that follows the control-flow with minimal β value is calculated by the function *getBestTransitiveAugmentation(…)*. It calculates all transitive augmentation solutions according to Definition 30 and returns a solution with the minimal γ value. If no transitive solution exists, all direct augmentation solutions are computed according to Definition 31 with the procedure $getAllDirectAugmentations()$. Obviously, multiple such solutions can exist, since multiple partners participating in the path can be in possession of the variable. We choose one solution for which the influence on the γ measure is minimal.

Algorithm 1 solution computeSingleSol(p, P, mode, r, v, c, o, inPos)

```
1: if checkPosession(o,r,v,c,P,inPos) then
2:     return p
3: end if
4: sol = getBestTransitiveAugmentation(p,P,mode,r, v, c, o)
5: if sol exists then
6:     return sol
7: else
8:     B = getAllDirectAugmentations(p,P,mode,r, v, c, o)
9:     return sol ∈ {d ∈ B : ∀d′ ∈ B : γ(d) ≤ γ(d′)}
10: end if
```

6.3.2 Computing an augmentation of a process

After having discussed the augmentation of a single augmentation path, we now present how an augmentation for an entire process is obtained by incrementally augmenting paths. The basic principle is shown in Algorithm 2.

In a first step, the set of all augmentation candidates is computed (according to Definition 28) and stored in the relation oR containing tuples of the form *(r, v, c, o, p)*. Each tuple represents that o is an origin of activity r for variable v, if the path p from o to r is executed. The condition c is the condition of the path (see Definition 27).

Then all nodes reading from variables of the input process P are traversed in a topological order. For each reader r, the algorithm iterates over every variable v read by r and every origin o of v for r. For every combination of reader r, origin o and variable v all paths between o and r are obtained from oR and one solution candidate is computed for every path using Algorithm 1. All solutions are stored in the set $CAND$. These solutions are not yet implemented in the augmentation and they are used as an estimate for the costs for their individual implementation.

While the set of candidate solutions $CAND$ is not empty, a solution $mySol$ in $CAND$ is chosen. We choose a solution with the minimum α value that requires the minimum number of communication steps. If multiple such solutions exist, one with the minimal γ value is chosen.

Then the selected solution $mySol$ is recomputed in order to take all communication steps that were previously implemented into account. The application of the solution to the augmentation according to Definition 33 is realized by the method *implementSolution(sol,o,v,P,inPos)*. The method gets the solution sol, the origin o, the variable v, the process P and the $inPos$ relations as input. It inserts the required communication steps of sol to the process P and updates the $inPos$ relation. For updating $inPos$, it adds one tuple (o, v, sr) for every inserted communication step sr. Finally, $mySol$ and all paths in $CAND$ that are in conflict (see Definition 36) with $mySol$ are removed from $CAND$.

As a postprocessing step, the equivalence transformation rules from [18] are applied to provide further optimizations. Most importantly, multiple communication steps with equivalent senders and receivers and equivalent variables are merged to single communication steps with a condition that is a disjunction of the conditions of the merged nodes (Rule TS6). Additionally, communication steps with the same actors and same conditions but different variables are merged (Rule TS5) and communication steps are re-arranged using rule $TS1b$.

6.3.3 Properties of the algorithm

Theorem 5 (Soundness) *Every augmentation P' of an input process with correct data-flow P generated by Algorithm 2 is correct according to Definition 9.*

Proof 5 In the first phase, the algorithm incrementally applies augmentation solutions for all Augmentation Candidates (see Definition 28) of the process where an augmentation is required. The resulting pre-solution is correct according to Theorem 3. The second phase of the algorithm (line 28) optimizes the correct augmentation produced in the first phase by applying equivalence transformations, which are proven to be correct [18].

Theorem 6 (Non-redundant augmentations) *Every augmentation P' of an input process with correct data-flow P provided by Algorithm 2 is non-redundant (see Definition 23)*

Proof 6 The proof directly follows from Theorem 4.

Algorithm 2 generateAugmentation(P, mode)

```
1: oR ← computeOriginRelation(P)
2: Let inPos be a relation of the form (o, v, sr) // o is a step, v is a variable, sr is a communication step.
3: for all r ∈ P.N in a topological order do
4:   for all v ∈ r.r do
5:     for all o ∈ {t.o|t ∈ oR ∧ t.v = v ∧ t.r = r} do
6:       CAND = new set of tuples (sol, p)
7:       for all (p, c) ∈ {(t.p, t.c)|t ∈ oR ∧ t.v = v ∧ t.r = r ∧ t.o = o} do
8:         pathSol = computeSingleSol(p, P, mode, r, v, c, o, inPos)
9:         CAND.add((pathSol, p))
10:      end for
11:      while CAND ≠ {} do
12:        mA = {t ∈ CAND|∀t′ ∈ CAND : α(t.sol) ≤ α(t′.sol)}
13:        mAB = {t ∈ mA|∀t′ ∈ mA : #cSteps(t.sol) ≤ #cSteps(t′.sol)}
14:        mABG = {t ∈ mAB|∀t′ ∈ mAB : γ(t.sol) ≤ γ(t′.sol)}
15:        Let mySol ∈ mABG
16:        mySol.sol = computeSingleSol(mySol.p, P, mode, r, v, c, o, inPos)
17:        implementSolution(mySol.sol, o, v, P, inPos)
18:        CAND = CAND\{mySol}
19:        for all oSol ∈ CAND do
20:          if inConflict(mySol.p,oSol,p) then
21:            CAND = CAND\{oSol}
22:          end if
23:        end for
24:      end while
25:    end for
26:  end for
27: end for
28: optimizeByRules(P)
29: return P;
```

Theorem 7 (Compliance with safeness classes) *Every augmentation P' of any input process with correct data-flow P provided by Algorithm 2 complies with the given safeness class.*

Proof 7 The proof follows directly from the definition of the augmentation graph (Definition 29) and the definition of transitive augmentations (Definition 30). The augmentation graph does by definition not contain edges with weight > 0 that are not allowed by the safeness class. Therefore, no solution can contain communication steps that do not adhere to the safeness class. Direct augmentation solutions (see Definition 31) are trivially strong dynamically safe. The applied transformation rules $TS1b$, $TS5$ and $TS6$ do not change the receivers of variables.

7 Evaluation

We have performed a comprehensive evaluation of our heuristic algorithm in order to assess the following evaluation goals.

– *EG1* How do the generated solutions of the heuristic approach compete with the base-line approaches [19] in terms of measure scores?

Table 1 Characteristics of the topologies for the different classes: minimum, maximum and average number of gateways, and number of activities

Class	Min # GW	Max # GW	Avg # GW	# Activities
S	6	10	7	16
M	10	16	12	50
L	22	26	24	100

– *EG2* How do the dimensions topology, size, number of actors, number of activities reading from and writing to variables influence the achievable measure scores?
– *EG3* Does the approach scale to provide reasonable performance for relevant classes of real world processes?

7.1 Data-set

For assessing EG1, we have evaluated all operation modes (fully transitive (H-UN), static needs (H-SN), weak dynamic (H-WDN) and strong dynamic (H-SDN)) of our heuristic approach against the base-line approaches fully transitive (B1), direct late send (B2) and direct early send (B3) [19]. We used a data-set containing 22,500 process models. It allows us to evaluate the influence of the size and topology of process models, the number of actors, the number of activities reading from and writing to variables on the achievable measure scores (evaluation goals $EG2$ and $EG3$). The data-set is composed of three top-level classes $small$, $medium$ and $large$ (see Table 1 for details). For each process class, there are 10 randomly generated process topologies. For each topology, there are 3 classes with different numbers of actors (S: 2,3,4 actors, M: 5,10,15 actors, L: 10,20,30 actors). For each such class, there are 10 random assignments of actors to process nodes. Finally, for each process there are 25 processes for all combinations of 20%, 40%, 60%, 80%, 100% readers and 20%, 40%, 60%, 80%, 100% writers. Each process model contains one variable to assess the influence on the measure for different numbers of readers and writers in isolation. This results in a total number of $3 * 10 * 3 * 10 * 5 * 5 = 22{,}500$ process models. The whole data-set, a documentation, all generated augmentations, additional figures, and all measurements are available online at http://isys.uni-klu.ac.at/PDF/prep/DatasetGeneratingAugmentations17.zip.

7.1.1 Overall measure scores

Table 2 summarizes the average measure scores we obtained running the algorithms over all the 22,500 process models. The comparison between $B1$ and $H-UN$ allows us to assess how beneficial tunneling is for reducing the overall number of data transmissions. Both algorithms augment a process in a way such that all the actors subsequent to a variable writer W are recipients of the value written by W. The difference is that $H - UN$ produces augmentations which are non-redundant by exploiting tunnels. Tunnels allow to avoid transmitting multiple times the same variable version to the

Table 2 Average values of α, β, and γ for the execution of the different algorithms over the 22,500 processes of all classes

Algorithm	α	β	γ
B1	0.00	54.54	0.00
H-UN	0.00	12.16	12.94
H-SN	0.97	10.10	11.61
W-WDN	2.76	7.60	11.18
H-SDN	3.10	7.28	11.37
B2	4.81	7.28	10.68
B3	19.55	23.59	6.35

same actor, thus reducing the overall number of transmissions. Our experiments show that the reduction in the value of β is considerably high, being on average 78% (from 54.54 to 12.16). Since the application of tunnels still realizes transitive transmissions, the value of α is 0. On the other hand, as tunneling potentially introduces the need to know additional decision variables, $H-UN$ brings on average an increase of γ from 0 to 12.94.

It is important to note that γ is not a weighted number. Weighting it by the number of process actors instead shows the increase in the number of decision variables that need to be communicated to each actor to realize the data-flow. The average number of actors per process in the complete set of processes is 11: augmentations generated by $H-UN$ require on average $12.94/11 = 1.18$ additional decision variables to be transmitted to each process actor. The γ values induced by other restriction classes of the heuristic approach (11.61, 11.18, 11.37) are very close to the one induced by $B2$ (10.68), and their weighted counterparts are even closer as they differ at most by $(11.61 - 10.68)/11 = 0.09$. The impact of the decision variable transmissions induced by the heuristic algorithm becomes then less significant in comparison to the base-lines, being on average higher by a negligible quantity.

Both $B2$ and $H-SDN$ provide solutions which are non-redundant and that belong to the same safeness class. Only the actors who really need to read a specific variable get its value. While $B2$ produces augmentations in which a variable is transmitted to a reader immediately before the reading step, with $H-SDN$ it is possible to use some man-in-the-middle to realize a transitive transmission. In any case no actor who does not need that variable version is involved. For this reason, both algorithms produce solutions with the very same β value. The advantage of $H-SDN$ is evident when we look at the α values for the generated solutions. In our experiments $H-SDN$ provided results with a better integration of data- and control-flow, which can be seen in an average α reduction of 36% (from α=4.81 to α=3.10). Concerning γ, our heuristic approach reaches values which are very close to the ones of the base-line (γ=11.37 for $H-SDN$, γ=10.68 for $B2$).

7.1.2 Influence on classes: measure scores per class

Table 3 shows the measure scores of the augmentations given by the different algorithms executions over processes of classes S, M, and L. The trends in the advantages

Table 3 Average values of α, β, and γ for the execution of the different algorithms over the 22,500 processes of classes S, M, and L

Algorithm	S class			M class			L class		
	α	β	γ	α	β	γ	α	β	γ
B1	0.00	16.17	0.00	0.00	53.54	0.00	0.00	93.91	0.00
H-UN	0.00	1.82	2.23	0.00	12.49	17.53	0.00	22.17	19.05
H-SN	0.09	1.72	2.13	0.83	10.91	16.27	1.99	17.68	16.43
H-WDN	0.24	1.58	2.12	2.59	8.50	15.57	5.46	12.71	15.85
H-SDN	0.27	1.55	2.13	3.03	8.08	15.77	5.99	12.22	16.21
B2	0.73	1.55	2.08	5.00	8.08	14.72	8.72	12.22	15.25
B3	2.29	3.86	0.79	18.44	23.07	9.81	37.93	43.83	8.45

of the heuristic approach versus the base-line approaches seen in the overall analysis are observed also in the single classes. In particular, when comparing $H-UN$ against $B1$, we observe a lower achieved value of β in the S class from 16.17 to 1.82 (86% less), in the M class from 53.54 to 12.49 (77% less), and in the L class from 93.91 to 22.17 (76% less). A lower value of α when comparing $H-SDN$ against $B2$ is achieved in all classes: from 0.73 to 0.27 (63% less) in the S class, from 5.00 to 3.03 (39% less) in the M class, and from 8.72 to 5.99 (31% less) in the L class.

7.1.3 Influence of number of actors, readers and writers, and safeness classes

To understand under which situations, we can expect the minimum and maximum benefits from the heuristic approach, it is interesting to observe the influence of the number of actors, readers, and writers over the measure scores. For a correct analysis, it is necessary to consider the process classes separately, as they differ in the number of actors. We concentrate our discussion on class M, since, due to its size and number of actors, we believe it is the most representative of real-world processes. Here we refer to $M05$, $M10$, and $M15$ as the subclasses of M induced by the different number of actors assigned to its processes. The same observations that we make on this class apply to the measures of the other classes, and to the overall measures as well. We now analyze the impact of the variations in the numbers of actors, readers and writers by comparing $H-UN$ and $H-SDN$ against their competing base-line approaches $B1$ and $B2$; subsequently we analyze the influence of the safeness classes over the scores of α and β.

1. *Comparison of $H-UN$ against $B1$* Comparing $H-UN$ against $B1$, for processes in $M05$, a minimum β advantage of 64.97% and a maximum of 92.68%, with an average of 82.11% were achieved. For processes in *M10*, a minimum β advantage of 54.25%, and a maximum of 89.81%, with an average of 73.38% were achieved. For processes in *M15*, the β advantage spans from a minimum of 51.75% to a maximum of 83.68%, with an average of 69.08%. We observe that when the number of actors is low, the β advantage of $H-UN$ compared to $B1$ is higher; as the number of actors increases the β advantage is lower.

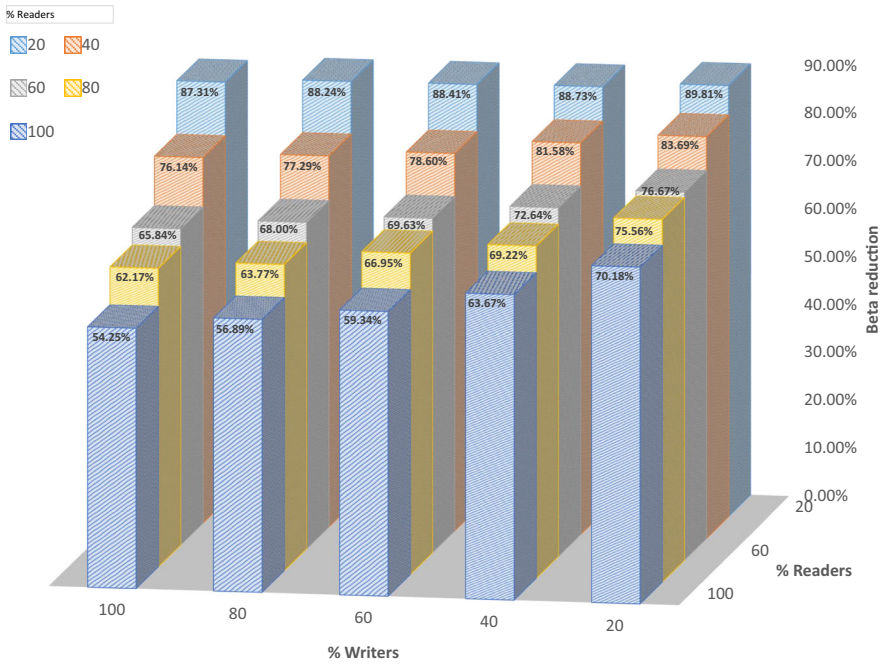

Fig. 7 Average values of β reduction from $B1$ to $H-UN$ for the processes of class *M10*

For processes in $M05$, in the broad range of 40% readers to 80% readers and 20 to 100% writers, the advantage in β ranges from 68.97% to 92.68%. For 20% readers, the β advantage spans from 89.05% to 92.02%. The lowest advantages are those relative to processes with 100% writers, but still good results are achieved, with a minimum β advantage of 64.97%. For processes in *M10*, as can be seen in Fig. 7, in the broad range of 40% to 80% readers and 20% to 100% writers, the advantage in β spans from 60.41% to 85.04%. For 20% readers, we observe an advantage in β from 87.03% to 89.93%. As for *M05*, also for this class the lowest β advantage is found in the case of 100% writers, with a minimum of 54.25%. For processes in *M15*, in the broad range of 40% to 80% readers and 20% to 100% writers very good average advantages in β of 61.26% to 75.50% were achieved. The case of 20% readers resulted in average advantage of the β value of 79.59% to 83.7%. The extreme case of 100% writers strongly reduces the utility of exploiting tunnels. Tunnels can only occur between split and join nodes but not between activities. However, still reasonable advantages of β of 51.75% to 79.51% were achieved. An overall trend is that the higher the number of writers or readers, the lower the advantage of our heuristic algorithm. However, the minimal average advantage in β is still 64.97% for *M05*, 54.25% for *M10*, and 51.75% for *M15*. Similar trends can be observed in the other classes and at global level.

2. *Comparison of* $H-SDN$ *against* $B2$ The comparison of $H-SDN$ against $B2$ shows an average advantage in the score of α of 50.05% for $M05$, of 33.04% for $M10$, and of 26.61% for $M15$. The same observation made in the comparison of

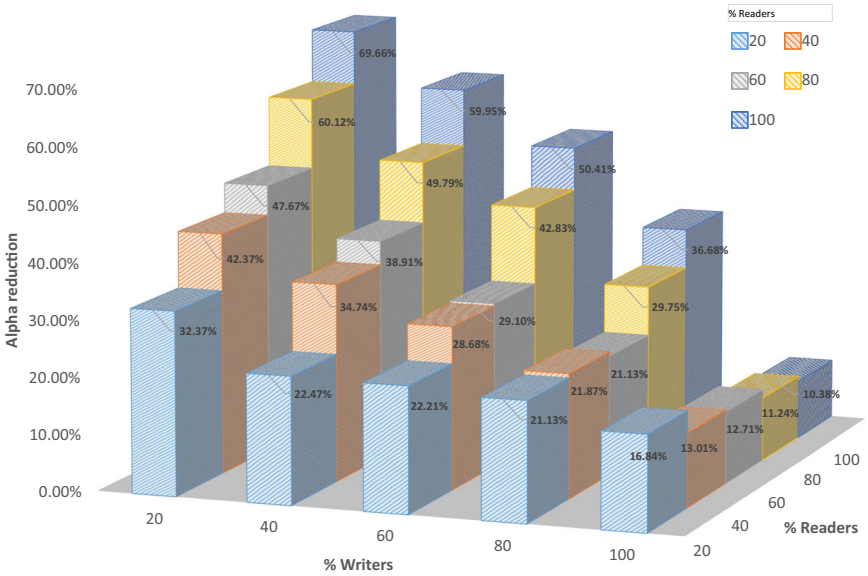

Fig. 8 Average values of α reduction from $B2$ to $H-SDN$ for the processes of class *M10*

$H-UN$ against $B1$ is valid also for the α advantage of $H-SDN$ against $B2$: less actors mean a higher chance to exploit tunnels and realize transitive transmissions (higher α advantage), while more actors reduce the likelihood of this opportunity and result in lower α advantage.

We now take a closer look at the advantages trend as the number of readers and writers changes. A clear trend we can observe is that the more readers there are in a process, the higher the α advantage for $H-SDN$ (Fig. 8). In particular we see that for *M05*, *M10* and *M15* the advantage is significant in the range 80 to 100% readers, with peaks of 75.84%, 69.66%, and 64.42%, respectively (averages over the combinations of writers for 100% readers: 55.38%, 45.41%, 40.43%). On the other hand, more writers lead to worse advantage. This is clearly visible in the 100% writers case, independently from the number of readers. The advantage induced by tunnels is almost all lost, being 29.21% (*M*05), 12.84% (*M*10), and 7.51% (*M15*) the worst α advantages (average over the different ratios of readers).

3. *Influence of safeness classes on α and β values* Figure 9 shows the general trends of the α and β values over the processes of class *M10* as the restriction level of the heuristic algorithm changes. We observe the increase in α (0.0, 0.80, 2.88, 3.34) as the number of allowed process actors for a transitive transmission increases, as well as the reduction of β (13.31, 11.78, 8.86, 8.40) which occurs at the same time. These trends are not surprising, since they are a consequence of the fact that by increasing the restriction level from $H-UN$ to $H-SDN$ the number of process actors that can be invoked for a variable transmission is more and more reduced. A lower number of possible actors determines, in turn, a lower chance of being able to exploit transitivity and tunnels, thus forcing more direct late transmissions, which means higher α. At the same time, the reduced number

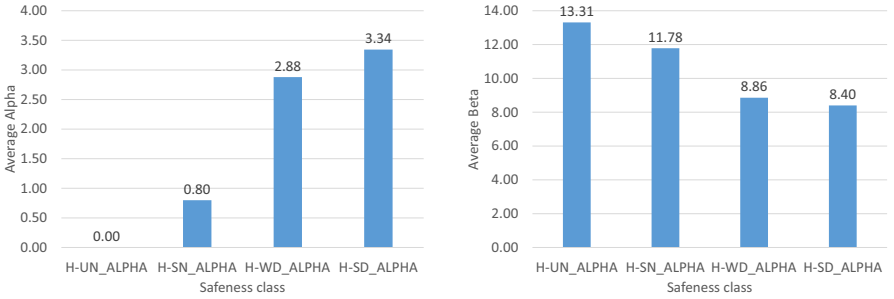

Fig. 9 Average values of α and β across the safeness classes for the processes of class *M10*

Table 4 Average and maximum execution times for the the different safeness classes of the heuristic algorithm over all the processes of class *L*

Algorithm	Avg time [s]	Max time [s]	95-percentile [s]
H-UN	0.70	212.22	3.10
H-SN	0.66	972.66	2.75
H-WDN	1.92	270.80	5.00
H-SDN	2.61	462.31	6.17

of involved actors means also a reduction in the number of unnecessary variable transmissions to actors who do not require the variable, therefore a lower β.

7.1.4 Scalability

Our third evaluation goal (*EG3*) aims at assessing the scalability of the heuristic approach for inter-organizational processes. Each process model used in the evaluation can be a global process model in an inter-organizational setting, and each task in it might be a sub-process in the local process model.

Table 4 reports the average and maximum times for the executions of the different restrictions of the heuristic algorithm over the processes in the *L* class, which was the most demanding in terms of computation. In 95% of these processes the execution of $H - UN$ required less than 3.10 s, the execution of $H - SN$ less than 2.75 s, the execution of $H - WDN$ less than 5.00 s, and the execution of $H - SDN$ required less than 6.17 s. The maximum times of execution were registered with processes characterized by the lowest number of writers and highest number of readers, since this combination results in the highest number of potential transmission paths. The maximum observed execution times are anyways below 8 min. The average memory usage for the execution of the heuristic over a process is 70.89MB. Since the generation of the data-flow implementation takes place at process design time, we conclude that our approach is applicable, both in terms of computation times and in terms of resource requirements. Our experiments therefore show that the heuristic approach scales for inter-organizational settings, since the executions were feasible for global processes with up to 100 activities (*L* class).

7.2 Discussion

1. The comparison between $H - UN$ and $B1$ shows that a significant reduction in β is achieved by the heuristic approach. Our experiments showed that the overall advantage is on average 78% (with a minimum average over processes in $M15$ of 51.75%, and a maximum average over processes in $M05$ of 92.68%). An average advantage of 78% in β means that $H - UN$ produces augmentations that on average require 78% less weighted communication steps than $B1$. The price to pay for the reduction of β is given by the increased γ value of $H - UN$, which, when weighted to count the number of additional decisions to communicate to each actor, is still very low. The heuristic approach proves therefore to be beneficial when the requirement is *data-flow follows control-flow*, since it avoids redundancies and keeps the number of additional decision variables to transmit to each actor low.
2. The comparison between $H - SDN$ and $B2$ shows that the corresponding β values are the same, since they add no redundant communications; however, $H - SDN$ reaches on average a 36% better integration of data- and control-flow (with minimum average over all processes in $M15$ of 26.61%, and a maximum over all processes of $M05$ of 50.05%). Even when the α advantage is minimal (7.51% in $M15$, in the case of 100% writers) no other measure is negatively affected, thus the heuristic approach still remains a viable choice for augmentation. Our experiments therefore show that $H - SDN$ is always more favorable than $B2$, as it allows better integration of data- and control-flow.
3. The scalability evaluation showed the practical feasibility of the heuristic approach, which was able to provide augmentations, even for big inter-organizational processes with 100 public steps and dozens of gateways, in less than 8 min in the worst case, and in less than 6 s in 95% of cases.

8 Related work

Early works on splitting a global process model into processes for each actor of a collaboration were presented in [12] for state charts and [3,7] for subclasses of Petri nets. They do not address the implementation of the data-flow perspective via messages. Ref. [12] assumes the existence of central data-stores, while [3,7] abstract from the data-perspective. In [7] the notion of accordance is introduced to formally describe whether a local process model is in accordance with the generated view of the global process model. However, this also only addresses the control-flow perspective.

Later approaches for splitting global processes into local processes for each actor of a collaboration were presented in [12–15,20,21]. We have presented an approach to generate local process models from a global process model with arbitrary assignments of activities to actors in [13]. The work addresses data-flow by generating communication steps for the distribution of decisions. However, the distribution of general case-variables is not addressed. In [14] and the follow up paper [15] a role based decomposition approach for BPEL processes is presented. The paper deals with BPEL specific problems, such as dead path elimination. The proposed solution for exchanging data between actors boils down to the late send base-line approach where

data is sent from the origin to the reader immediately before invoking the reader. An alternative approach for splitting processes which is not bound to a specific modeling language is presented in [20]. It presents approaches for splitting different workflow patterns. The implementation of data-flow between partitions is very briefly discussed and a solution equivalent to early direct send is sketched. More details are given in [21]. In particular, activities writing to a variable that is later read by an activity of another actor are connected by message exchange between the writing and the reading activity. In [26] another perspective is taken into account. There, an approach is proposed that focuses on privacy requirements, allowing to define which of the participants are allowed to exchange messages and which are not: if certain exchanges are forbidden, then an alternative solution is generated if possible. In contrast to our heuristic solution presented in this paper, the previously discussed approaches either ignore [3,7] data-flow, apply base-line strategies [14,15,20,21] or provide only one single solution [26] for the implementation of inter-organizational data-flow.

However, as we could show in [18], there are far more correct solutions for implementing data-flow between actors. The work in [18] presents a sound and complete set of equivalence transformations that allow to derive any correct data-flow implementation from any other correct data-flow implementation. The work thus spans the solution space for all potential data-flow implementations. Our work in [19] has introduced measures for augmentations (which were refined in this paper) and provides an evaluation of the measure characteristics of the base-line approaches. We complete our previous works with an efficient algorithm that allows to generate optimized augmentation solutions adhering to any safeness class.

The previously discussed works abstract from actual message contents or their format. A work addressing this issue was presented in [27]. It extends the original public to private approach from [3] with global and local data models and corresponding mappings. The model-driven approach allows to derive the required message contents and formats between actors. The exchange of data itself is realized through message exchanges, which start from the writer and end at the reader. However, the way the message exchange is actually implemented is abstracted, both for the global choreography and for the local process models. In contrast, we derive optimized, concrete implementations of message exchanges but we abstract from data contents of process variables.

Another broad stream of research on process partitioning deals with decentralizing a previously centralized process with the aim of optimizing the performance with works such as [9,28–31]. In these works, the assignment of activities to actors is not realized by the designer. Instead, the approaches derive optimal assignments based on various criteria. This strongly differs from our work, where we generate optimized implementations of the data-flow perspective via message exchanges for a given assignment of actors to activities. However, optimizing partitions / actors assignments to tasks is also a highly complex problem demanding for efficient implementations such as the heuristic approach presented in [28] or the approach in [30] which employs a genetic algorithm.

Approaches that emphasize more on the design perspective of inter organizational data-flow were presented in [32,33]. Ref. [32] provides a method to probabilistically derive sets of public data-items for each business party participating in an inter-organizational

process. The work abstracts from the actual implementation of message exchanges and determines what shared data-elements are required for each path in the global process. These sets may be used by designers to actually implement the data-exchange. In contrast, our approach automates the implementations following various restriction classes.

In Monsieur et al. [33] a pattern language for the design of data-flow in coordination scenarios, in which different participants carry on a process is proposed. The work distinguishes between transmission and request patterns and direct and indirect patterns for data-flow initiation. While the work aims in guiding designers to select the best pattern for their specific scenario, our approach automatically generates direct or indirect data-flow implementations via message exchanges depending on the required safeness class. In addition to the mentioned indirect data transmission pattern, [33] provides some evaluation criteria for the choice between direct and indirect transmission of data in a process, such as data confidentiality and data reusability. These criteria also apply for selecting a proper safeness class.

Our approach aims in generating communication steps that realize the data-transfer between the participants in an optimized manner. An alternative solution is to employ a middleware approach, where all communicating actors communicate via a middleware which is run in a distributed way as proposed in [34,35]. While the work in [35] mentions the potential of data-flow optimizations within the middleware, no specific optimization approaches are introduced. Middleware systems add an additional layer of abstraction and can reduce the complexity in modeling inter-organizational processes with data. We address the same issue by automatically deriving the required message exchanges at build-time rather than to require a modeler to define them manually. This allows to execute the processes at runtime without making assumptions on the software of the cooperating actors. In contrast to [34,35], only means to send and receive data are required. No additionally middleware is needed.

9 Conclusion

Automatically generating good implementations of inter-organizational data-flow implementations via message exchange is a very complex task. We reflected on the notion of a 'good' implementation and propose a set of three measures which can be used to discuss certain partially conflicting qualities of implementing the data-flow and which correspond to qualities relevant in practice. Existing approaches apply static strategies with fixed patterns when and from whom data is sent to other actors requiring the data. Such static strategies lead to rather mediocre data-flow implementations in terms of measure scores, in particular we show that they optimize one aspect at the expense of the other aspects.

Based on these results, we have developed a heuristic algorithm that generates good and non-redundant solutions for a large set of user preferences. Our comprehensive experimental evaluation shows that the heuristic algorithm provides in average significantly better measure scores over existing solutions. Even for worst-case scenarios, our implementation still produces significantly better implementations of the data-

flow in terms of measure scores than the approaches proposed so far. The algorithm is scalable and thus also feasible for large inter-organizational processes.

The results of our work in combination with control-flow partitioning approaches [13] allows to automatically derive optimized choreographies from global process models instead of requiring designers to manually model message exchanges between organizations.

The work reported here is part of an ongoing endeavour to improve the handling of data in inter-organizational processes. The results we obtained from this work encourage us for further extending the process model expressiveness by including additional control patterns with the ambition to support a wide range of process definition languages used in practice. In this paper we focus on optimizing data-flow implementations in terms of required interactions, overall band-width requirements and decision overhead. Notwithstanding these rather technical criteria, there are other quality criteria like maintainability or understandability, which have to be taken into account for designing and implementing inter-organizational processes. Integrating these factors with the metrics proposed here is subject of future work. Other addressed questions include the collaboration of process enactment services built on different paradigms (activity centric, artifact centric, declarative, etc.) and the extension of case handling standards for inter-organizational collaboration.

Acknowledgements Open access funding provided by University of Klagenfurt.

References

1. van der Aalst, W.M.P.: Process-oriented architectures for electronic commerce and interorganizational workflow. Inf. Syst. **24**(8), 115–126 (1999)
2. Groiss, H., Eder, J.: Workflow systems for inter-organizational business processes. ACM SIGGroup Bull. **18**, 23–26 (1997)
3. van der Aalst, W.M.P., Weske, M.: The p2p approach to interorganizational workflows. In: CAiSE '01, pp. 140–156. Springer, New York (2001)
4. Norta, A., Eshuis, R.: Specification and verification of harmonized business-process collaborations. Inf. Syst. Front. **12**(4), 457–479 (2010). https://doi.org/10.1007/s10796-009-9164-1
5. Chiu, D.K., Cheung, S., Till, S.: Workflow view driven cross-organizational interoperability in a web service environment. Inf. Technol. Manag. **5**, 221–250 (2004)
6. Chebbi, I., Dustdar, S., Tata, S.: The view-based approach to dynamic inter-organizational workflow cooperation. Data Knowl. Eng. **56**(2), 139–173 (2006). https://doi.org/10.1016/j.datak.2005.03.008
7. van der Aalst, W.M.P., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty contracts: agreeing and implementing interorganizational processes. Comput. J. **53**(1), 90–106 (2010)
8. Knuplesch, D., Pryss, R., Reichert, M.: Data-aware interaction in distributed and collaborative workflows: modeling, semantics, correctness. In: CollaborateCom, pp. 223–232 (2012)
9. Chafle, G.B., Chandra, S., Mann, V., Nanda, M.G.: Decentralized orchestration of composite web services. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, WWW Alt. '04, pp. 134–143. ACM, New York (2004). https://doi.org/10.1145/1013367.1013390

10. Eshuis, R., Grefen, P.: Constructing customized process views. Data Knowl. Eng. **64**(2), 419–438 (2008). https://doi.org/10.1016/j.datak.2007.07.003
11. Schulz, K.A., Orlowska, M.E.: Facilitating cross-organisational workflows with a workflow view approach. Data Knowl. Eng. **51**(1), 109–147 (2004). https://doi.org/10.1016/j.datak.2004.03.008. Contact-driven coordination and collaboration in the Internet context
12. Wodtke, D., Weissenfels, J., Weikum, G., Dittrich, A.K.: The mentor project: steps towards enterprise-wide workflow management. In: Proceedings of the 12th International Conference on Data Engineering, pp. 556–565 (1996). https://doi.org/10.1109/ICDE.1996.492206
13. Köpke, J., Eder, J., Künstner, M.: Top-down design of collaborating processes. In: Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services, iiWAS '14, pp. 336–345. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2684200.2684282
14. Khalaf, R., Leymann, F.: E role-based decomposition of business processes using bpel. In: ICWS '06, pp. 770–780 (2006). https://doi.org/10.1109/ICWS.2006.56
15. Khalaf, R., Kopp, O., Leymann, F.: Maintaining data dependencies across bpel process fragments. In: Service-Oriented Computing ICSOC 2007, *LNCS*, vol. 4749, pp. 207–219. Springer (2007). https://doi.org/10.1007/978-3-540-74974-5_17
16. Eder, J., Lehmann, M., Tahamtan, A.: Choreographies as federations of choreographies and orchestrations. In: International Conference on Conceptual Modeling, pp. 183–192. Springer, New York (2006)
17. (OMG), O.M.G.: Business process model and notation (bpmn) version 2.0. Tech. rep. (2011)
18. Köpke, J., Eder, J.: Equivalence transformations for the design of interorganizational data-flow. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) Advanced Information Systems Engineering: 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8–12, 2015, Proceedings, pp. 367–381. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-19069-3_23
19. Köpke, J., Franceschetti, M., Eder, J.: Analyzing data-flow implementations for distributed execution of inter-organizational processes. In: Proceedings of IIWAS 17 (in print), iiWAS '17 (2017)
20. Fdhila, W., Godart, C.: Toward synchronization between decentralized orchestrations of composite web services. In: CollaborateCom 2009, pp. 1–10 (2009). https://doi.org/10.4108/ICST.COLLABORATECOM2009.8275
21. Fdhila, W., Yildiz, U., Godart, C.: A flexible approach for automatic process decentralization using dependency tables. In: IEEE International Conference on Web Services, ICWS 2009, Los Angeles, CA, USA, 6-10 July 2009, pp. 847–855. IEEE Computer Society (2009). https://doi.org/10.1109/ICWS.2009.41
22. van der Aalst, W.M.P., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**(1), 5–51 (2003)
23. van Der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: Business Process Management. Springer, New York (2000)
24. Combi, C., Gambini, M.: Flaws in the flow: The weakness of unstructured business process modeling languages dealing with data. In: OTM 2009, *LNCS*, vol. 5870, pp. 42–59. Springer, New York (2009). https://doi.org/10.1007/978-3-642-05148-7_6
25. Trka, N., van der Aalst, W.M.P., Sidorova, N.: Data-flow anti-patterns: Discovering data-flow errors in workflows. In: CAiSE '09, *LNCS*, vol. 5565, pp. 425–439. Springer, New York (2009). https://doi.org/10.1007/978-3-642-02144-2_34
26. Yildiz, U., Godart, C.: Information flow control with decentralized service compositions. In: ICWS 2007, pp. 9–17 (2007). https://doi.org/10.1109/ICWS.2007.109
27. Meyer et al., A.: Automating data exchange in process choreographies. In: CAiSE 14, *LNCS*, vol. 8484, pp. 316–331. Springer, New York (2014). https://doi.org/10.1007/978-3-319-07881-6_22
28. Fdhila, W., Dumas, M., Godart, C., García-Bañuelos, L.: Heuristics for composite web service decentralization. Softw. Syst. Model. **13**(2), 599–619 (2014). https://doi.org/10.1007/s10270-012-0262-z
29. Goettelmann, E., Fdhila, W., Godart, C.: Partitioning and cloud deployment of composite web services under security constraints. In: 2013 IEEE International Conference on Cloud Engineering, IC2E 2013, San Francisco, CA, USA, March 25–27, 2013, pp. 193–200. IEEE Computer Society (2013). https://doi.org/10.1109/IC2E.2013.22
30. Ai, L., Tang, M., Fidge, C.: Partitioning composite web services for decentralized execution using a genetic algorithm. Futur. Gener. Comput. Syst. **27**(2), 157–172 (2011). https://doi.org/10.1016/j.future.2010.08.003

31. Esfahani, F.S., Murad, M.A.A., Sulaiman, M.N.B., Udzir, N.I.: Adaptable decentralized service oriented architecture. J. Syst. Softw. **84**(10), 1591–1617 (2011). https://doi.org/10.1016/j.jss.2011.03.031
32. Guo, X., Sun, S.X., Vogel, D.: A dataflow perspective for business process integration. ACM Trans. Manag. Inf. Syst. **5**(4), 22:1–22:33 (2014). https://doi.org/10.1145/2629450
33. Monsieur, G., Snoeck, M., Lemahieu, W.: Managing data dependencies in service compositions. J. Syst. Softw. **85**(11), 2604–2628 (2012). https://doi.org/10.1016/j.jss.2012.05.092
34. Hahn, M., Karastoyanova, D., Leymann, F.: Data-aware service choreographies through transparent data exchange. In: Bozzon, A., Cudre-Maroux, P., Pautasso, C. (eds.) Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings, pp. 357–364. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-38791-8_20
35. Hahn, M., Breitenbücher, U., Leymann, F., Weiß, A.: Trade—a transparent data exchange middleware for service choreographies. In: H. Panetto, C. Debruyne, W. Gaaloul, M.P. Papazoglou, A. Paschke, C.A. Ardagna, R. Meersman (eds.) OTM 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I, *LNCS*, vol. 10573, pp. 252–270. Springer, New York (2017). https://doi.org/10.1007/978-3-319-69462-7_16