Semantic Normalization and Merging of Business Dependency Models

Preliminary Version

Alexander Motzek*, Christina Geick*, Ralf Möller*
* Universität zu Lübeck, Institute of Information Systems, Germany.
Email: motzek@ifis.uni-luebeck.de, christina.geick@student.uni-luebeck.de, moeller@uni-luebeck.de

***Abstract*—Assessing potential threats and impacts relevant for a company, requires a detailed analysis of a company's business processes and functions down to a level of infrastructure resources, in the form of one business dependency model. Required information is frequently encapsulated in BPMN models per process, but pose an eminent problem of fusing and merging multiple sources into one model. Experts defining BPMN models possibly use different nomenclature, descriptions, and references towards common entities, leading to semantically overlapping partial dependency models. Merging multiple partial dependency models is a novel problem related to the business process matching problem, but origins from an orthogonal perspective. In this paper we propose a business dependency model normalization and matching approach by exploiting structures and dependencies of business resources, which neither requires linguistic processing nor "fuzzy" matching processes.**

I. Introduction

Understanding and assessing potential impacts relevant for a company or mission is a pertinacious problem and a novel research area. For example, a local impact caused by an event on a distant node, say, user workstation, might lead to a causal chain of operational failures, leading to an impact on a company, as some critical devices where affected required in critical business processes. In order to understand which processes and resources of a company are critical, must be protected, and to what extent they represent criticality, various approaches, for example, by Jakobson [1] or Motzek [2] employ a dependency analysis of a company. They decompose a company into its business processes, depending on tasks requiring information from data stores. In essence, it must be understood which processes are critical to a company and which (groups of) devices are required *directly* during all business processes. It is a common practice, e.g., proposed by Barreto [3], Musman [4] or Motzek [2] to extract such information from BPMN models by an extraction of involved entities in each modeled process. On top of that, we show that such dependency models are further designable by experts.

In order to assure a correct impact assessment, *one* solid, consistent business dependency model of a company, capturing the entire scope of an organization is required. In effect, duplicate entities in a model would lead to incorrect impact assessments. When given information from multiple sources, e.g., multiple BPMN models and information gathered from different experts from different expertises, a common problem is eminent: Experts frequently use different nomenclatures, descriptions, languages and references for common entities, inevitably leading to semantic overlaps between information sources due to semantic and terminology gaps. Therefore, a naive concatenation of extracted and gathered business dependency model inevitably leads to duplicate entities and incorrect impact assessments. To obtain well-defined results, i.e., to obtain a solid and consistent business dependency model from multiple sources and experts, a semantic normalization and merging is required for business dependency model. In the scope of this paper we position and relate problems around business dependency models between "classical" business process matching, ontology matching and graph isomorphism problems, and consecutively propose a solution to the semantic normalization and merging problem among business dependency models.

The business process matching problem is well investigated, for example by Dijkman [5], but solutions, as we outline throughout this article, are not directly applicable towards business dependency model matching. Approaches required to match dependency models are also related towards ontology matching (cf. [6]), which, however, frequently rely on natural language processing. We identify two problems, leading to an assumption that existing approaches are not directly applicable: In a real world use case, descriptions and identifiers of entities were sounding so similarly, s.t., any linguistic matching is bound to fail. Further, the business dependency model matching problem is less restrict than the business process matching problem. In order to emphasize the latter, we present the following example: Two modeled business processes, e.g., two BPMN models, do not match exactly, but a dependency analysis still leads to an exactly matching model. For example, a different ordering of requests, e.g., to a webserver and a database, is likely to represent two different (but, partially matching) business processes, but does not lead to an exact match. However, both processes are dependent on the web- and database server to the same extent, and both dependency models will match exactly. We say that a dependency analysis is performed orthogonally to a processing analysis. Nevertheless, we believe that in most cases an ordering of tasks in a business process is only marginally relevant to solve the (partial) business process matching problem and that our decomposition approach can help to solving the business process matching problem as well. As discussed by, e.g., Dijkman [5] or Shvaiko [6], both, process matching and ontology matching, are known to be very hard problems. In this article we present a dependency-model matching and merging approach that scales linearly with the number of to-be-processed dependency models and number of nodes in a model.

The contribution of this paper can be summarized as follows: This paper discusses matching and merging problems associated with business dependency models and relates emerging problems towards similar problems among business process models. Without introduction of "fuzzy" or approximate matching techniques, we present an approach that solves the exact business dependency model matching problem and scales well with the number of to be matched models and that provides a working base for future partially matching approaches. We believe that the presented approach is applicable towards the (partial) business process matching problem as well.

The remainder of this paper is structured as follows: In Sec. II we introduce a business dependency model and an associated use case. We formalize the business dependency model matching and merging problem in Sec. III and present an approach to solve the matching and merging problem. We experimentally evaluate and discuss our approach in Sec. IV and conclude in Sec. V.

II. Business Dependency Models

An impact assessment is used to address potential impacts onto a higher goal, from widespread events which lead to local impacts. For example, a local impact caused by an event on a distant node, might lead to a causal chain of operational failures, leading to an impact on a company. Understanding these impacts is a pertinacious problem and current work uses adhoc solutions based on handcrafted algorithms. While such approaches deliver early results, their assessments need to be verified and validated by large amounts of data—which is not always available.

Motzek et al. introduce an approach towards impact assessment based on a probabilistic graphical model in [2], which defines a well-understood problem, namely, probabilistic inference in graphical models, on which an assessment can be reduced. By resorting to a probabilistic model, the use of conditional probability distributions allows for a local view on assessments, without a need to understand a specific use case nor any algorithmic properties. It is this local view, which allows a validation of defined data. This means that assessments from experts can be used directly without global normalization factors and experts are not forced outside their expertise. By the use of multiple probabilistic models, expertise from different experts, namely, business, operational and security experts, are combineable in one large probabilistic graphical model for mission impact assessment. For example, an expert in charge of assessing a business dependency analysis must not reason about operational dependencies of, say, a critical webpage on a complex computation cluster and data warehouse. Likewise, an operational expert assessing dependencies inside computational clusters, must not reason about which devices are directly critical and which contribute passively to critical devices—a distinction often not uniquely identifiable.

However, Motzek et al. [2] base their approach on an expert's design of a dependency model for a company. An expert is not always available or, maybe, does not even exist. Still, as they argue as well, such information is likely to be already present in the form of BPMN models, as also suggested by, e.g., Barreto [3]. It is straightforward from the definition of their business (or mission) dependency models on how such an extraction is performed. Motzek et al. [2] extend a model by Jakobson [1] and model mission dependencies as shown in Fig. 1 as a graph of *mission nodes* (MN). A *company* is dependent on its *business processes*. A business process is dependent on one or more *business functions*, which are provided by *Business resources*. Fig. 1 shows a dependency graph of business relevant objects for a small company consisting of two business processes, requiring a total of four functions provided by four resources. Relations to an original BPMN model for BP_1 (refer Fig. 2) are straightforward, where each BPMN model represents one business process, a BPMN-task represents a business function and BPMN-data stores (and references) represent business devices.

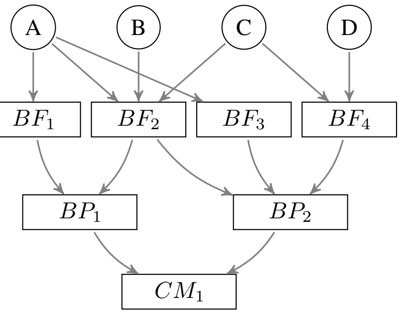

Figure 1. A mission dependency model representing a small company CM_1 with two identified critical business processes, transitively dependent on critical resources A, B, C and D. If BP_1 and BP_2 are given separately, then BF_2, A, B and C are present multiple times, share possibly ambiguous identifiers, and must be matched and merged. *(CPDs not shown)*

In fact, the mission dependency model represents a probabilistic graphical model and degrees of dependencies are modeled as local conditional probability distributions. An ordering and sequence of nodes in a BPMN model is exploited towards obtaining assessments of these distributions. For example, a parallel path can provide information that a redundancy in tasks and devices is present. We argue that degrees of dependencies are not relevant for the business dependency model matching problem.

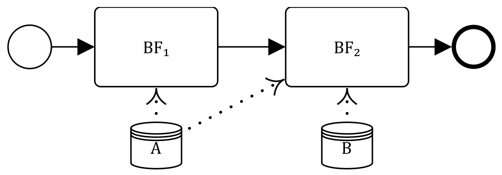

Figure 2. Example BPMN 2.0 model sketch for the BP_1 business process shown in the dependency model of Fig. 1.

For the scope of this paper we formally define a business dependency model from a graph perspective.

Definition 1 (Business dependency model)**.** *A business dependency model (MG) is a directed acyclic graph (DAG) as a pair $\langle \vec{V}, \vec{E} \rangle$ of vertices V and edges E. Vertices $\vec{V}$ are categorized according to their semantic as business- devices ($\vec{BD}$), -functions ($\vec{BF}$), -processes ($\vec{BP}$), and -company (BC). For the scope of this work we consider that a business* *dependency model is created for a single BC. The ordering $BD \prec BF \prec BP \prec BC$ represents the topological ordering of graph MG. Every edge $E \in \vec{E}$ represents a dependency. Each vertex $V \in \vec{V}$ is associated with attributes ID, $Name$, $Description$, for which we write V^a when referring to an attribute a of V. Let $V \in \vec{V}$ and let $\vec{E}_v \in \vec{E}$ be the set of edges directed to V, then let $\vec{d}_V$ be the set of vertices from which $\vec{E}_v$ origin, i.e., $\vec{E}_v$ is the set of dependencies of V. For every vertex $V \in \vec{V}$ a conditional probability distribution (CPD) $P(V|\vec{d}_V)$ is given.* ▲

For the scope of this work, we solely consider inter-layer dependencies, and exclude intra-layer dependencies, e.g., we exclude dependencies of a business function onto another business function. We argue that such dependencies are resolvable in a lower level and by an adequate specification of associated CPDs.

In fact, as outlined in more detail in Sec. IV, we perform impact assessment as described by Motzek [2] in a real world use case for a large industrial company, Acea SpA, Italy's largest water services operator and one of the largest energy distribution companies in Italy [7]. Information was not given in the form of BPMN models, and business processes were described to us by multiple business experts, as well as, IT experts, by word of mouth. Fusing all information into one model required to match different descriptions, names and languages to the same entities. In particular, some experts used host names to describe ICT resources, where others used IPs. Further, sometimes business functions where named arbitrarily as an abstract conglomeration of resources, whereas another expert described valuable information to be included in the description of a business function and a business process. Additionally, abbreviations were used frequently and were taken for granted, requiring careful matching between, e.g., "HVC" and "High Level Voltage Control". We believe that these problems are highly likely to be evident as well, if experts create BPMN models and an automatic extraction is performed.

As outlined before, matching business processes is not sufficient for matching dependency models, for which the following section formally discusses a solution.

III. Merging and Matching Problems

A business dependency model is required to obtain a view onto a company to assess potential impacts. If a dependency model contains multiple entities twice, an impact is accounted for multiple times during an assessment leading to spurious results. As sketched in the previous section, a transformation from a BPMN model towards a dependency model is straightforward by considering solely the set of modeled BPMN-tasks without their sequential orderings. However, once multiple BPMN models or partial dependency models are processed, sets of data stores (business devices) and tasks (business functions) are likely to overlap semantically and must be merged towards one unified model.

Linguistic approaches, such as string matching algorithms to identify common entities by their description, name or ID however where bound to fail in our use case. For example, in our use case, business functions, were labeled "BF-HVD", "BF-HVC", "BF-MVD" and "BF-MVC", which represented completely different functions, but involved partially matching references to business devices. While in a classic BPMN modeling approach, the sequential ordering and structure of tasks delivers sufficient information to exclude false positive linguistic matches, the softening of structure in dependency models does not allow a linguistic matching anymore.

We, therefore, present an approach for solving business dependency model matching problem based on a pure structural analysis of the dependency model and an existing inventory of addressable resources. In the following we carefully differentiate between multiple matching and merging problems in order to clearly define when two models "match" and how a "merging" should be executed. We present an approach for solving a specialized problem and experimentally evaluate that it solves the semantic normalization and merging problem.

Definition 2 (Semantic normalization and merging problem, SMP)**.** *Let m_1, m_2 be mission dependency models. Then, the semantic business dependency model normalization and merging problem (SMP) is the problem to obtain m from m_1 and m_2 without a duplication of nodes and vertices that bear an identical semantic and to normalize all vertices in m into a standardized format.* ▲

Informally, a solution to the SMP shall represent how a human would merge and normalize two presented dependency models. Essentially, the SMP is only solvable directly by an expert, as a clear definition of "identical semantic" is missing. To obtain a clearer definition, we define the following.

Definition 3 (Business dependency model matching problem, BDMP)**.** *Let m_1, m_2 be mission dependency models and let mn be a vertex of an arbitrary mission dependency model. Then the BDMP is the problem to find the set $\vec{mn}$, s.t., for all $mn \in \vec{mn}$ holds $\forall mn \in \vec{mn} : \exists (mn_1 = mn) \in m_1 \wedge (mn_2 = mn) \in m_2$ with respect to an equality operator $=$.* ▲

The simplest definition of an equality operator, would be an exact equality, for which we write $=_*$. Namely, for $=_*$ to hold, all attributes and dependencies of two nodes must exactly match. As argued before, we doubt that, due to multiple experts, designers and naming schemes, a matching problem based on $=_*$ is of any avail. Still, it is indisputable that two dependency models that match exactly ($=_*$) are seen as semantically identical by a human as well. Further, we assume that if two nodes share a unique ID, i.e., share a non-trivial (randomly) generated unique identifier, written $=_{id}$, one usually assumes semantic equality as well. This means that if an expert trusts that IDs are unique, an experts performs a *semantic* matching on IDs. We will therefore develop an approach that firstly normalizes all nodes to unique IDs, s.t., the BDMP w.r.t. $=_{id}$ does represent a semantic match as performed by an expert, i.e., represents the SMP.

Definition 4 (Identity equality)**.** *Let mn_1, mn_2 be two arbitrary mission nodes. Let $\vec{mn}$ represent the set of* all *possible $\vec{mn}$ existing. Then mn_1 and mn_2 are identity equal, written $mn_1 =_{id} mn_2$, iff, $mn_1^{id} = mn_2^{id}$ and $\nexists mn_3 \in \{\vec{mn} \backslash mn_1, mn_2\} : mn_3^{id} = mn_1^{id}$. Two sets of mission nodes*

$\vec{mn}_1$, $\vec{mn}_2$ are identity equal, if every node in $\vec{mn}_1$ is identity equal to exactly one node in $\vec{mn}_2$ and vice versa. ▲

We use the definition of identity equality in order to reduce the SMP problem to a more tangible problem: If the BDMP for $=_{id}$ represents the SMP, then the problem that needs to be solved is a normalization, s.t., we uniquely identify individual entities. This means that not complete graphs must be matched, but only single entities. As demonstrated later, this significantly reduces computational complexity.

To normalize all nodes to be identified by unique IDs, we exploit that data stores, i.e., business devices are not only referenced by business models, but are part of an infrastructure and are referenced by, e.g., large inventories, i.e., other independent sources. While in ICT-related use cases such inventories can be created automatically by network mappers and scanners, in other use cases, e.g., in logistics, such inventories are present as well in the form of stock lists or equipment lists. We therefore obtain identity equality by normalizing all business devices based on inventory equality.

Definition 5 (Inventory equality)**.** *Let bd_1, bd_2 be business devices and let I be an inventory. Then bd_1 and bd_2 are inventory equal w.r.t. I, written $bd_1 =_I bd_2$, if an non-empty arbitrary attribute of bd_1 is equal to an arbitrary non-empty attribute of exactly one resource $g \in I$ and an arbitrary non-empty attribute of bd_2 is equal to an arbitrary non-empty attribute of g.* ▲

For the scope of this article and our use case, we use attributes representing IP-addresses (IPv4 and 6), MAC-addresses, hostnames and unique node identifier. During conversations with experts from our use case we found that devices were most often referenced by their hostname and (static) IP address. Extensions to inventory equality, e.g., will involve a partial matching of hostnames or, to some extent, a match of IP ranges.

As mentioned before, such inventories only exist for physically existing devices and do not cover modeled tasks or processes themselves. Still, as one is now able to uniquely identify dependencies of tasks (business functions) onto business devices, we argue that two functions depending on the same set of devices must be semantically identically as well. We therefore define structural equality.

Definition 6 (Structural equality)**.** *Let mn_1, mn_2 be mission nodes. Let $\vec{d}_{mn}$ represent the set of dependencies of a mission node mn. Then mn_1 and mn_2 are structurally equal, written $mn_1 =_s mn_2$, iff, $\vec{d}_{mn_1} =_{id} \vec{d}_{mn_2}$ and $\vec{d}_{mn_1} \neq \emptyset$.* ▲

For the scope of this work we only consider an exact structural equality, i.e., both dependencies sets must match completely. It is straightforward to extend this work to apply partial structural matching by defining a difference metric between both sets $\vec{d}_{mn_1}$ and $\vec{d}_{mn_2}$ and allowing a degree of deviation. A modification of our approach towards partial matching is later given as a corollary on future work. Still, once a degree of deviation is allowed in structural matching, an extent of "fuzzyness" is introduced which might lead to false positives. The following corollary sketches a combination of attribute and structural matching to prevent false positive partial structural matching.

Corollary 1 (Structural equality linguistic sufficiency conditions)**.** *So far attributes are not considered, and, attributes are deemed to be equivalent given a structural match. Still, there is the possibility that semantically different business functions or processes, in fact, produce the same dependency model, or are almost identical, if one allows for a degree of deviation. In such a particular situation, a structural equality will lead to a false-positive merging, but could be prevented by using attributes as sufficient conditions. For example, two business functions, named "Credit Card Fraud Detection" and "Customer Balance", will both directly dependent on a customer database and a transaction history. If a degree of deviation in dependencies is allowed, it is likely that both will be structurally identical. A simple linguistic comparison can prevent a false positive match in this case. Due to the assumption that structural equality already identifies a high degree of (or even a a complete) semantic similarity, a linguistic comparison must only be performed as a false-positive prevention on the set of structurally equivalent nodes, and can be less restrictive. If a linguistic comparison would be employed as a first-step classification, a function "Credit Card Fraud Detection" would not be matched with an (otherwise structurally identical) process named "CCFD".*

We therefore envision to use attributes solely as a plausibility check. To do so, we group structural identical nodes ($Struct_i$), as shown in Fig. 3 as a sorted collect of structurally identical nodes inside them, which again are grouped by leading attributes that shall be used for a linguistic comparison. This is sketched as an example for a leading attribute of the name in Fig. 3. This has the benefit that post a structural matching, a plausibility linguistic matching can be performed efficiently and can be extremely unrestrictive. For example, a descriptive attribute can be seen as being linguistically identical, once a certain degree of non-trivial words are found. ▲

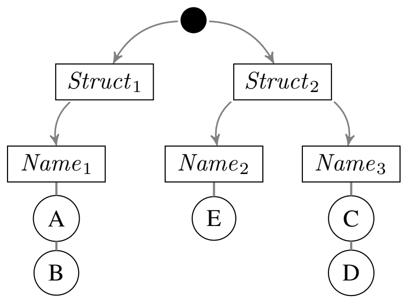

Figure 3. Structural model to perform a postponed linguistic sufficiency check on structurally identical nodes. $A - E$ represent business functions (or, respectively, processes), grouped per structural identity ($Struct_i$). Structurally equivalent nodes are grouped by (partially) matching leading attributes used for linguistic matching purposes; in this example $Name$.

In summary, we obtain a clear definition of when ($=_{id}$) and how (exploiting $=_I$, $=_s$) models are seen as semantically equivalent. Using these definitions, we propose an algorithm as presented in Alg. 1 that firstly enforces identity equality on business devices through exploitation of inventory equality, and consecutively creates identity equality on business functions by exploitation of structural equality $=_s$. Respectively, identity equality is obtained among business processes through exploitation of structural equality on business functions.

Algorithm 1 Incremental match and merge

```
1: procedure MATCH-AND-MERGE($m_1$ to $m$, given $I$)
2:   for all $BP \in (\vec{BP} \in m_1)$ do
3:     for all $BF \in (\vec{BF} \in m_1)$ do
4:       for all $BD \in (\vec{BD} \in m_1)$ do
5:         $BD \leftarrow BD_{norm} : BD_{norm} =_I BD$
6:         $m \leftarrow m \cap BD$
7:       $BF \leftarrow BF_{norm} : \{(BF_{norm} \in m) =_s BF\} \vee BF$
8:       $m \leftarrow m \cap BF$
9:     $BP \leftarrow BP_{norm} : \{(BP_{norm} \in m) =_s BF\} \vee BP$
10:    $m \leftarrow m \cap BP$
11:  return $m$
```

Informally, the Alg. 1 firstly searches and normalizes each business device in a supplied inventory I (depending on the definition of $=_I$ a partial match is defined as well). Based on the normalized devices, which are added to the target model m, structural matches of business functions in m_1 according to $=_s$ are searched in m. Unmatchable business functions are appended to the target model m and business processes of m_1 are normalized and matched respectively. For efficiently finding a structural matching entity (Lines 7 and 9) we use Alg. 2 which encodes a dependency structure as a unique string representation. Alg. 2 creates structure-identifying strings, which are natively sorted by increasing numbers of dependencies. That all reference strings are sorted ascending by increasing size of dependencies is achieved prefixes generated in Line 4. If fixed-length identifiers for dependencies are used, a structurally equivalent mission node can be found by a binary search in the set of reference strings.

Algorithm 2 Dependency string encoding

```
1: procedure ENCODE-STRUCTURE(mission node $mn$)
2:   if $mn \in \vec{BD}$ then return $mn^{ID}$
3:   $\vec{d_{mn}} \leftarrow sort(\vec{d_{mn}})$ by $|\vec{d_d}|, d \in \vec{d_{mn}}$
4:   $C \leftarrow |\vec{d_{mn}}|-$
5:   for all $d \in \vec{d_{mn}}$ do
6:     $C \leftarrow C \cap$ ENCODE-STRUCTURE($d$)
7:   return $C$
```

Example 1 (Structural string encoding)**.** *To demonstrate the structural encoding identifier, we refer to Fig. 1. A structural encoding for BF_1 is* `1-A` *and for BF_2 is* `3-ABC`*. Then, a structural encoding of BP_1 is* `2- 1-A 3-ABC` *(white-space are added for better readability). Likewise, one obtains* `3- 1-A 2-DC 3-ABC` *for BP_2*

Note that if no fixed-length identifiers are used, further delimiters are required, e.g., for BP_1 like `2-(1-(A)).(3-(A).(B).(C))`*.* ♦

Given a dependency model m consisting of n_{BP} business processes, n_{BF} referenced functions and an inventory consisting of n_{BD} devices, to which a new dependency model m_1 with n^*_{BF} and n^*_{BD} is to be merged, Alg. 1 and 2 perform a structural match in $n^*_{BD} \cdot \log n_{BD} + n^*_{BF} \cdot \log n_{BF} + \log n_{BP}$, if one uses a binary search to find matching devices and dependency structures. Note that the nested loops in Alg. 1 do traverse the topological ordering of the new dependency model m_1 and normalize entities bottom-up, i.e., all entities of m_1 are solely processed once.

We argue that an inventory equivalence and structural equivalence must represent a semantic equivalence, and one can assign unique IDs. Therefore, one obtains a solution to the SMP:

Theorem 1. *The procedure represented by Alg. 1 solves the BDMP with respect to a semantic equivalence.* ▲

This theorem is proven experimentally in the following section. Further, the structural encoding can be extended towards partial matching of business process dependencies, which we outline in the following corollary.

Corollary 2 (Partial dependency matching)**.** *We exploit structural equivalence, i.e., we exploit that two entities that are dependent on the same dependencies are seen as identical. If, for example, nine out of ten dependencies match, one could argue that both entities are still seen as identical. The encoding to reference structures used by Alg. 1 bears significant advantages for such a partial structural match, and can be seen as a string alignment problem. Say, one is given two string-encodings for business processes by* `CD ABC DEF` *and* `Q ABC CDE DEG` *(excluding the dependency size prefixes). An approach to align both dependency structures is fairly similar to computing a Damerau-Levenshtein-Distance between both. We envision the usual four modification patterns as insertion, deletion, substitution and transposition, where transposition is only allowed on complete business-function blocks (delimited by white-spaces) but is cost-free. Then, the first dependency structure is aligned to the second by substituting* `F` *and* `G` *in* `DEF`*, adding* `Q`*, transposing* `CD` *by one, and adding* `E` *to* `CD` *with an edit-score of* 3 *(transpose is free).*

Notwithstanding, a partial alignment can also be performed on a deeper encoding level, i.e., for matching business functions directly. However, then information about partial matches is eliminated too early. It is beneficial to perform this alignment on business processes, as one gains further information: If one merges `CDE` *and* `CD` *too early, i.e., during matching of business function, one loses the information that, in fact,* `CD` *and* `CDE` *are used by two higher-level partially matching business processes. For example, if two functions* `CDE` *and* `CD` *are used by two to-be-matched business processes which are otherwise completely different, it is less likely that functions* `CDE` *and* `CD` *match. In contrary, if both functions are used by otherwise completely identical business processes, it is more likely that the functions match. By performing an alignment at a business processes level, one preserves and can incorporate this knowledge.* ▲

As usual, given a matching set between two models, it is the common problem to unify all information towards one model.

Definition 7 (Business dependency model merging problem, BDFP)**.** *Let m_1, m_2 be business dependency models and let $\vec{mn}_=$ be a solution to the BDMP between m_1,m_2 w.r.t. $=$.*

Then the BDFP is the problem to transfer all attributes of pairs $\langle mn_1, mn_2 \rangle \in \vec{mn}_=$, where $mn_1 = mn_2$ into a unified node mn_n. ▲

Essentially, if all matching nodes are not exactly equivalent, lossy and lossless merging operators exist. As business devices are normalized through inventory equality, business devices immediately are exactly equivalent and no merging operation needs to be executed. Still, merging different descriptions and names of, e.g., business functions remains as a problem. Nevertheless, to perform an impact assessment merged descriptions and names are not required. We therefore, for the scope of this article, use a simple lossless merging approach.

Proposition 1 (Lossless business dependency model merging)**.** *Given a BDFP for models m_1, m_2 and $\vec{mn} =$, a lossless exact solution is to transfer each $\langle mn_1, mn_2 \rangle \in \vec{mn}_=$ into mn_n by concatenating all attributes.* ▲

Note that, as we use an exact structural match, only attributes need to be merged and that dependencies are equivalent per definition.

In summary, our approach performs a structural match between two dependency models and scales linearly with the number of to be matched and merged dependency models. A pure merge based on structural equivalence is possible, as ground truth is incorporated by consulting external sources where individual assets of modeled processes are referenced as well. In the following section we experimentally evaluate our approach using models derived from a real world use case, artificial duplication and heavy modifications.

IV. Experiments and Discussion

We implement and apply a probabilistic mission impact assessment as presented in [2] in a large industrial enterprise, Acea SpA, Italy's largest water services operator and one of the largest energy distribution companies in Italy [7]. In this use case a complete business dependency model was created by hand by us originating from different descriptions of experts from different expertise in the enterprise. We obtained four business processes, requiring seven business functions provided by 26 business devices, where all dependencies are deeply meshed, as a high degree of redundancy is present in the industrial network. Effectively, large sets of business functions and associated devices overlapped in the requirements of each business process. In order to test our approach we extract a partial dependency model for every business process from our model and artificially manipulate it, preserving structures, but manipulating almost all semantic information. By doing so, we simulate common errors that would occur when extracting partial dependency models from individual BPMN models. Attributes of business devices are only modified to an extend that they are still identifiable by $=_I$. To be precise, we randomly remove unique identifiers and manipulate, switch and remove description- and name- attributes, and randomly add words, letters and punctuation. In this evaluation, the number of total business processes remains constant. Results, shown in Fig. 4, show that the total running time scales linear with the total amount of processed dependency model, i.e., the time to perform a match and merge is constant.

To show that our approach scales linear with the number of merged business processes, we randomly generate $10\,000$ business processes by random combinations of the enterprise's business functions. Every generated business processes is modified ten times by the aforementioned process. Therefore, we match and merge $100\,000$ partial dependency models representing $10\,000$ unique processes and denote the total running time over the number of acquired unique business processes in Fig. 5, i.e., one obtains 10 measurements for every business process. As evident from Fig. 5, the approach scales linearly with the number of unique business processes, i.e., performs a match and merge in constant computation time. Due to an apparently very low factor of lookup time of existing dependency encodings, a factor of $\log n_{BP}$, (n_{BP}: the number of known business processes, in which a dependency encoding must be found) was not measurable by us. In fact, we were able to process $100\,000$ partial dependency models in less than seven minutes on a standard Intel Core i5-4300U mobile CPU.

Further, these performance evaluations greatly prove the semantic normalization and merging performed by Alg. 1 as stated by Thm. 1:

Proof of Theorem 1. After all evaluations, i.e., after processing $120\,000$ models, the obtained merged dependency model included exactly the same business processes (including the artificially added), the same business functions and the same business devices as the original model and the generated graph is isomorph to the original model, i.e., all dependency structures were preserved and no entities were removed or duplicated. Therefore, the presented approach based on structural and inventory equivalence indeed solves the SMP for the generated sets, as stated by Thm. 1, under the in-place modifications. □

All generated dependency models are manipulated to an extent, where any linguistic matching would fail and resemble commonly made errors. In fact, a structural match is the only left matching ability, even for an expert in our generated test set. Furthermore, considering that (partial) business process matching is a very hard problem [5] and that our approach scales linearly is a promising result. It is likely that two matching dependency models are based on two (partially) matching business process models and can, as well, be used as a preprocessing step for business process matching. Admittedly, the manipulation is chosen in a way such that inventory equality must be assured by our approach. If inventory equality is not assurable by our approach, i.e., neither hostnames nor unique identifiers such as IDs, IPs or MAC addresses are available, our approach will fail. Still, we highly believe that without any of these attributes, no expert could uniquely identify a common entity as well. Nevertheless, Col. 2 can allow one to partially restore such unmatchable entities in retrospect.

We motivate a view of business process in the form of business dependency models to perform an impact assessment, where dependency models are a required input data. Viewing companies, or most often missions, as dependency models decomposing an asset into multiple layers is as well done by Barreto et al. [3] who introduce a well-understood modeling

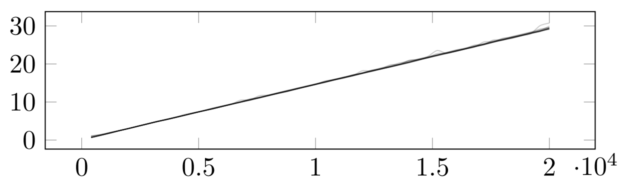

Figure 4. Total computation time (t_{mm}, ordinate in seconds) for incrementally matching and merging is linear within the number of dependency models n_{MG} (abscissa) merged so far. 13 evaluations over a set of 20 000 models displayed superimposed.

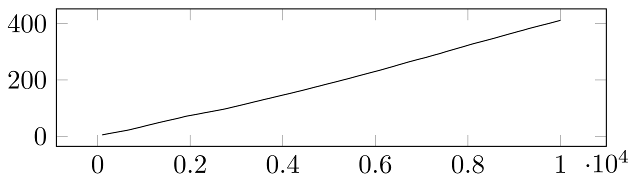

Figure 5. Total computation time (t_{mm} ordinate in seconds) for incrementally matching and merging is linear within the number of unique business processes n_{BP} (abscissa) obtained so far. Mean of 10 evaluations displayed.

technique and also use BPMN models to acquire knowledge, but do neither focus on nor discuss matching and merging of gathered information.

Albanese et al. present in [8] a well-modeled formalism for complex inter-dependencies of missions as a set of tasks, but do not consider the acquisition of such knowledge. Further, their models are tree-structured, i.e., accept duplication of multiple entities. Jacobson [1] presents a well understood conceptual framework using interdependencies based on operational capacity, his work is extended by Motzek et al. [2] who both view compies as a dependency graph. While Motzek et al. do discuss an acquisition of knowledge, Jacobson does not. Both do not consider an automatic extraction, matching and fusing of multiple information sources.

Various works presuppose modeled business processes, the underlying art of designing and managing business processes from organizational perspectives in the first place is especially discussed by Malone et al. in [9] and [10] and overlooking work [11] by Crowston et al.

Goodall et al. [12] and D'Amico et al. in [13] discuss mission modeling with the use of ontology based descriptions. Especially, Goodall et al. discuss fusion and data integration of multiple sources. However, as their work is based on a different conceptual level, the use of ontologies is not applicable to our intended use case of mission impact assessment. Still, the range of ontology merging and matching is a deep research area as well, frequently relying on natural language processing, which is not exploitable in the first place for our work. Being a broad field or research we refer to overlooking work by Shvaiko and Euzenat [6] and [14].

This article exploits identical dependency structures in a business dependency model in order to identify (potentially) semantic overlaps. Therefore, one can reduce the SNMP to a subgraph isomorphism problem with an exact isomorphism of labeled leafs, i.e., business resources. With such a reduction, one could modify existing solutions, e.g. initial work by Ullman [15] or novel advances by Cibej and Mihelic [16], but would then be prone to an (at least) NP-complete problem.

V. Conclusion

We introduce, discuss and formalize novel problems emerging around business dependency models, which are related to business process models, but are viewed from a orthogonal perspective. Where business process models intend to cover sequences of actions, dependency models try to grasp an overview of a company as a conglomerate of abstract, multiple concepts down to a deeper layer of resources. A view is required in order to understand possible impacts and threats to a company based on deeply meshed resource infrastructures, e.g., found in large industrial ICT and ICS focused companies.

We present, discuss and evaluate an approach to fuse and merge multiple sources of business dependency models into one consistent model for performing impact assessments. We apply our approach in a real world use case, demonstrate that the approach provides valuable starting points for extended work, and show that the approach scales well. Future work is dedicated to integrate partial dependency model matching, and is dedicated to research on the applicability of dependency model matching towards business process model matching. Moreover, we investigate more deeply on partial structural matching including linguistic plausibility checks. By considering allocations of larger partially-matching linguistic subgroups will allow for refining matched groups at higher levels.

Acknowledgements

This work was partly supported by the Seventh Framework Programme (FP7) of the European Commission as part of the Panoptesec integrated research project (GA 610416).

References

[1] G. Jakobson, "Mission Cyber Security Situation Assessment using Impact Dependency Graphs," in *FUSION 2011: 14th International Conference on Information Fusion, Chicago, Illinois, USA, July 5-8, 2011*, 2011, pp. 1–8.

[2] A. Motzek, R. Möller, M. Lange, and S. Dubus, "Probabilistic Mission Impact Assessment based on Widespread Local Events," in *NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, NATO IST-128 Workshop, Istanbul, Turkey, June 15-17, 2015*, 2015, pp. 16–22.

[3] A. de Barros Barreto, P. C. G. da Costa, and E. T. Yano, "Using a Semantic Approach to Cyber Impact Assessment," in *8th Conference on Semantic Technologies for Intelligence, Defense, and Security, Fairfax VA, USA, November 12-15, 2013*, 2013, pp. 101–108.

[4] S. Musman, A. Temin, M. Tanner, D. Fox, and B. Pridemore, "Evaluating the Impact of Cyber Attacks on Missions," in *5th International Conference on Information Warfare and Security*, 2010, pp. 446–456.

[5] R. M. Dijkman, M. Dumas, and L. García-Bañuelos, "Graph Matching Algorithms for Business Process Model Similarity Search," in *BPM 2009: 7th International Conference on Business Process Management, Ulm, Germany, September 8-10, 2009*, 2009, pp. 48–63.

[6] P. Shvaiko and J. Euzenat, "Ontology Matching: State of the Art and Future Challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 158–176, 2013.

[7] Acea SpA, "The Acea Group," 2016. [Online]. Available: http://www.acea.it/section.aspx/en/acea_spa

[8] M. Albanese, S. Jajodia, R. Jhawar, and V. Piuri, "Reliable Mission Deployment in Vulnerable Distributed Systems," in *DSN Workshops 2013: 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop, Budapest, Hungary, June 24-27, 2013*. IEEE, 2013, pp. 1–8.

[9] T. W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. S. Osborn, A. Bernstein, G. Herman *et al.*, “Tools for Inventing Organizations: Toward a Handbook of Organizational Processes,” *Management Science*, vol. 45, no. 3, pp. 425–443, 1999.

[10] T. W. Malone, K. Crowston, and G. Herman, *Organizing Business Knowledge: The MIT Process Handbook*. MIT Press, 2003.

[11] K. Crowston, J. Rubleske, and J. Howison, “Coordination Theory: A Ten-Year Retrospective,” *Human-Computer Interaction in Management Information Systems*, vol. 1, pp. 120–138, 2004.

[12] J. R. Goodall, A. D’Amico, and J. K. Kopylec, “Camus: Automatically Mapping Cyber Assets to Missions and Users,” in *Military Communications Conference*. IEEE, 2009, pp. 1–7.

[13] A. D’Amico, L. Buchanan, J. Goodall, and P. Walczak, “Mission Impact of Cyber Events: Scenarios and Ontology to Express the Relationships between Cyber Assets, Missions, and Users,” in *5th International Conference on Information Warfare and Security*, 2010, pp. 8–9.

[14] P. Shvaiko and J. Euzenat, “Ten Challenges for Ontology Matching,” in *OTM 2008: On the Move to Meaningful Internet Systems, Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Part II*, 2008, pp. 1164–1182.

[15] J. R. Ullmann, “An Algorithm for Subgraph Isomorphism,” *Journal of the ACM*, vol. 23, no. 1, pp. 31–42, 1976.

[16] U. Cibej and J. Mihelic, “Improvements to Ullmann’s Algorithm for the Subgraph Isomorphism Problem,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, no. 7, 2015.