Data & Knowledge Engineering xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Improved model management with aggregated business process models

H.A. Reijers [a,*], R.S. Mans [a], R.A. van der Toorn [b]

[a] School of Industrial Engineering, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
[b] ING Investment Management, Beatrixlaan 15, NL-2595 AK The Hague, The Netherlands

ARTICLE INFO

Article history:
Received 10 April 2008
Received in revised form 27 September 2008
Accepted 27 September 2008
Available online xxxx

Keywords:
Business process modeling
EPCs
Model management
Conceptual modeling
Workflow management
Application

ABSTRACT

Contemporary organizations invest much efforts in creating models of their business processes. This raises the issue of how to deal with large sets of process models that become available over time. This paper proposes an extension of event-driven process chains, called the *aggregate EPC* (aEPC), which can be used to describe a set of similar processes with a single model. By doing so, the number of process models that must be managed can be decreased. But at the same time, the process logic for each specific element of the set over which aggregation takes place can still be distinguished. The presented approach is supported as an add-on to the ARIS modeling tool box. To show the feasibility and effectiveness of the approach, we discuss its practical application in the context of a large financial organization.

1. Introduction

Modeling is at the core of both organizational design and information systems (IS) development [15]. In particular, the importance of *business process modeling* is suggested by many IS success factor studies, especially those reporting on large-scale multimillion dollar implementations, such as Enterprise Systems implementation projects [4,12,18,30].

One of the primary purposes of a business process model is to serve as *a means of communication*: it facilitates the understanding of a complex business process among various stakeholders [20,23,24,29]. Business process models may be used to this end much as an architect will use models to ascertain the views of users, to communicate new ideas, and to develop a shared understanding amongst participants [21]. Typical examples of stakeholders are managers, end-users, IT system developers, human resource staff, quality professionals, etc. Because business process models are increasingly used to demonstrate a company's compliance with various regulations (e.g. SOX, Basel II, UCITS) external authorities and accountants can be considered important stakeholders too.

The intensive use of business process models has its flip side: organizations are facing an ever-increasing burden of disclosing up-to-date business process models to their stakeholders. To appreciate the size of this maintenance effort, it should be noted that one project alone may result in the creation of dozens, hundreds or even thousands of business process models [6,17]. A single model in its turn may cover a dozen of departments within several legal entities, describe hundreds of steps, and relate to thousands of different agents with varying skills and responsibilities. Because many organizations are in a permanent flux of reorganization programs, introduce new products and services on a regular basis, and are subjected to new and changing legislations all the time, business process models need frequent updates. We will refer to the overall problem

* Corresponding author. Tel.: +31 40 247 36 29; fax: +31 40 243 26 12.
E-mail addresses: h.a.reijers@tue.nl (H.A. Reijers), r.s.mans@tue.nl (R.S. Mans), robert.van.der.toorn@ingim.com (R.A. van der Toorn).

0169-023X/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2008.09.004

of (1) maintaining a large set of business process models and (2) disclosing them in a way that is meaningful to a mixed audience of stakeholders as "business process model management".

This paper proposes a way to simplify business process model management. The key idea is that business process models can be combined into *aggregate* models, exploiting the commonalities between them. For example, consider two process models in a banking context, which capture the handling of loan applications from respectively corporate and private clients. Even though many of the checks to be executed *differ* for these different types of clients, the two process models may include a *similar* procedure for preparing and sending out the loan proposal. An aggregate model will combine both models into one, where the common part is included only once. In addition, it will contain some unique parts from each of the separate models that it is based on, which explicitly refer to the specific product, service or customer group they are concerned with. As a result of applying aggregate models, fewer models need to be maintained and updates to common parts will only need to be carried out once.

Of course, aggregate models will become larger and more complex than ordinary business process models. And because stakeholders cannot be expected to be modeling experts themselves [6], every effort must be made to give them access, through process models, to the information they desire. The way we propose to do this is to allow for the *on-demand* extraction of a process model from an aggregate process model that is specific for just a single type of product or service. For various purposes, such a *singular* model is exactly what a business user may want to see. In this paper, an algorithm is presented that implements this idea.

The presented approach to aggregate process models builds on the event-driven process chain (EPC) modeling language [19,38]. In comparison with some other modeling techniques, the EPC language is relatively easy to use by business people and the resulting models are fairly easy to understand [37]. Moreover, the modeling language is the core technique for depicting business processes within both the ARIS Toolset and the SAP R/3 system. These are the market leaders for process modeling and Enterprise Resource Planning, respectively. The presented approach is inspired by the business environment of ING Investment Management, a global asset manager, and implemented as an *add-on* to the ARIS Toolset that is used in this organization.

This paper is structured as follows. Section 2 provides some preliminaries, in particular the explanation of the EPC modeling language. In Section 3, we will present a list of requirements on aggregate models that we used to develop our ideas. After that, we will informally explore in Section 4 what the extraction algorithm should look like, which can spawn off singular process models from aggregate process models. Subsequently, in Section 5, the extraction algorithm itself is presented. In Section 6, we will show the feasibility of our approach by discussing its application within ING Investment Management and demonstrate how it has been integrated in the ARIS Toolset. This paper ends with a section on related work, a conclusion, and an agenda for future work.

2. Preliminaries

In this section, we introduce the preliminaries that are needed for the remainder of this paper. We will formally introduce some graph notions and the event-driven process chain (EPC) modeling language.

2.1. Graph theory

Definition 1 (Directed graph, path, cycle, connected, pre-set, post-set). A *directed graph* G is an ordered pair $G = (V, D)$:

- V is a finite, non-empty set of vertices (nodes).
- $D \subseteq (V \times V)$ is a set of ordered pairs of vertices (directed edges).

A *path* in a directed graph is a sequence of nodes $\langle v_1, v_2, \ldots, v_n \rangle$ such that for all $i, 1 \leqslant i < n$ $(v_i, v_{i+1}) \in D$. A *cycle* is a path where the start node and end node are the same. A directed graph is *connected* when for every two nodes $n, m \in V$ there is either a path from n to m or from m to n. With $\bullet n = \{m \in V | (m, n) \in D\}$, we refer to the predecessors of n in G (*pre-set*). Similarly, $n\bullet = \{m \in V | (n, m) \in D\}$ is the set of successors of n in G (*post-set*).

Definition 2 (Tree, leave, rooted tree, root, root path). A *tree* $G = (V, D)$ is a directed graph that is connected and contains no cycles. The nodes in a tree that have no successors are called *leaves*; the set of leaves in a graph G is denoted as *leaves*(G). A *rooted tree* is a tree with one special node $r \in V$, called the *root*, such that for every other node $v \in V$ there is a unique path from r to v. A path that leads from the root to a leaf is called a *root path*.

Note that we will overload the notions of path, pre-set and post-set in this paper for the graph-like structures as they appear within EPC's. From the context, it will become clear to which graph structure we refer.

2.2. Event-driven process chains

The EPC language was developed in 1992 at the Institute for Information Systems in Saarbrücken in cooperation with SAP AG [19]. The primary goal behind this development was to allow business users to describe processes on the level of their

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

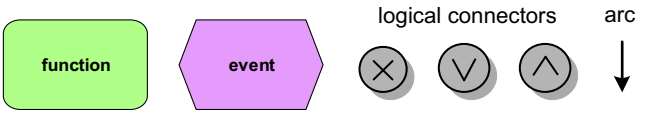

Fig. 1. The building blocks of an event-driven process chain.

business logic in a form that is easily understandable. A specific EPC model (or simply EPC) consists of the following building blocks:

- *Functions*: The basic building blocks are functions. A function corresponds to an activity (task, process step) which needs to be executed.
- *Events*: Events describe the situation before and/or after a function is executed. An event corresponds to the post condition of the function it succeeds (if any) and to the precondition of the function it precedes (if any).
- *Logical connectors*: Connectors can be used to show the different paths the process can take. Through the use of connectors, paths can be split and joined. There are three types of connectors. $\wedge$ (and), *XOR* (exclusive or) and $\vee$ (inclusive or).
- *Arcs*: Functions, events and connectors are connected by directed arcs.

The building blocks can be graphically represented as shown in Fig. 1.

In an EPC, the business process is given as a chain of events and functions.[1] The formal definition of an EPC can now be given as follows [19].

Definition 3 (Event-driven process chain). An event-driven process chain is a five-tuple (E, F, C, l, A):

- E is a finite (non-empty) set of events.
- F is a finite (non-empty) set of functions.
- C is a finite set of logical connectors.
- $l \in C \to \{\wedge, XOR, \vee\}$ is a function which maps each connector onto a connector type.
- $A \subseteq (E \times F) \cup (F \times E) \cup (E \times C) \cup (C \times E) \cup (F \times C) \cup (C \times F) \cup (C \times C)$ is a set of arcs.

From the specification of relation A in this definition it can be seen that it is not allowed to have an arc connecting two functions or two events. There are many more requirements an EPC should satisfy, e.g. only connectors are allowed to branch, there is at least one start event, there is at least one final event, and there are several limitations with respect to the use of connectors [1]. To formalize these requirements we build on additional concepts and notations from [35], in particular to distinguish between the various types of connectors.

Definition 4. $(N, C_\wedge, C_\vee, C_{XOR}, C_J, C_S, C_{EF}, C_{FE})$. Let $EPC = (E, F, C, l, A)$ be an event-driven process chain:

- $N = E \cup F \cup C$ is the set of nodes of *EPC*.
- $C_\wedge = \{c \in C | l(c) = \wedge\}$ is the set of AND connectors.
- $C_\vee = \{c \in C | l(c) = \vee\}$ is the set of OR connectors.
- $C_{XOR} = \{c \in C | l(c) = XOR\}$ is the set of *XOR* connectors.
- $C_J = \{c \in C || \bullet c| \geqslant 2\}$ is the set of join connectors.
- $C_S = \{c \in C || c \bullet | \geqslant 2\}$ is the set of split connectors.
- $C_{EF} \subseteq C$ is the set of connectors on a path from an event to a function, i.e. $c \in C_{EF}$ if and only if there is a path $p = \langle n_1, n_2, \ldots, n_{k-1}, n_k \rangle$ for $k \geqslant 3$ such that $n_1 \in E, n_2, \ldots, n_{k-1} \in C, n_k \in F$, and $c \in \{n_2, \ldots, n_{k-1}\}$.
- $C_{FE} \subseteq C$ is the set of connectors on a path from a function to an event, i.e. $c \in C_{FE}$ if and only if there is a path $p = \langle n_1, n_2, \ldots, n_{k-1}, n_k \rangle$ for $k \geqslant 3$ such that $n_1 \in F, n_2, \ldots, n_{k-1} \in C, n_k \in E$, and $c \in \{n_2, \ldots, n_{k-1}\}$.
- $C_{EE} \subseteq C$ is the set of connectors on a path from an event to an event, i.e. $c \in C_{EE}$ if and only if there is a path $p = \langle n_1, n_2, \ldots, n_{k-1}, n_k \rangle$ for $k \geqslant 3$ such that $n_1 \in E, n_2, \ldots, n_{k-1} \in C, n_k \in E$, and $c \in \{n_2, \ldots, n_{k-1}\}$.
- $C_{FF} \subseteq C$ (the set of connectors on a path from a function to a function) such that $c \in C_{FF}$ if and only if there is a path $p = \langle n_1, n_2, \ldots, n_{k-1}, n_k \rangle$ for $k \geqslant 3$ such that $n_1 \in F, n_2, \ldots, n_{k-1} \in C, n_k \in F$, and $c \in \{n_2, \ldots, n_{k-1}\}$.

These notations allow for the definition of the syntactical correctness of an EPC, as follows.

Definition 5 (Correctness of an event-driven process chain). An event-driven process chain $EPC = (E, F, C, l, A)$ is syntactically correct if and only if the following requirements are satisfied:

[1] Note that we abstract from additional features often found in tools supporting the modeling of EPCs, for example 'organizational units' and 'supporting systems'.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

- The sets E, F, and C are pairwise disjoint, i.e. $E \cap F = \emptyset, E \cap C = \emptyset$, and $F \cap C = \emptyset$.
- For each $e \in E : |\bullet e| \leqslant 1$ and $|e \bullet| \leqslant 1$.
- There is at least one event $e \in E$ such that $|\bullet e| = 0$.
- There is at least one event $e \in E$ such that $|e \bullet| = 0$.
- For each $f \in F : |\bullet f| = 1$ and $|f \bullet| = 1$.
- For each $c \in C : |\bullet c| \geqslant 1$ and $|c \bullet| \geqslant 1$.
- C_J and C_S partition C, i.e. $C_J \cap C_S = \emptyset$ and $C_J \cup C_S = C$.
- C_{EE} and C_{FF} are empty, i.e. $C_{EE} = \emptyset$ and $C_{FF} = \emptyset$.
- C_{EF} and C_{FE} partition C, i.e. $C_{EF} \cap C_{FE} = \emptyset$ and $C_{EF} \cup C_{FE} = C$.

The first requirement of this definition states that each component has a unique identifier. Note that connector names are omitted from the graphical representation of an event-driven process chain in a diagram. The other requirements correspond to restrictions on the relation A. Events cannot have multiple input arcs and there is at least one start event and one final event. Each function has exactly one input arc and one output arc. A connector c is either a join connector ($|c \bullet| = 1$ and $|\bullet c| \geqslant 2$) or a split connector ($|\bullet c| = 1$ and $|c \bullet| \geqslant 2$). The last requirement states that a connector c is either on a path from an event to a function or on a path from a function to an event.

Because each of the sets $\{C_J, C_S\}, \{C_{EF}, C_{FE}\}$, and $\{C_\wedge, C_{XOR}, C_\vee\}$ partitions C, one can theoretically imagine $2 \times 2 \times 3 = 12$ kinds of connectors. However, in the original definition of EPCs [19] two of these 12 combinations are not allowed: (i) A split connector of type C_{EF} cannot be of type *XOR* or type $\vee$, i.e. $C_S \cap C_{EF} \cap C_{XOR} = \emptyset$ and (ii) $C_S \cap C_{EF} \cap C_\vee = \emptyset$. As a result of this restriction, there are no choices between functions sharing the same input event. A choice is resolved *after* the execution of a function, not *before*. In the formalization of EPCs, we will not impose this restriction and consider $C_S \cap C_{EF} \cap C_{XOR} = \emptyset$ and $C_s \cap C_{EF} \cap C_\vee = \emptyset$ as a guideline rather than a strict requirement. This is also consistent with the modeling guidelines of ING Investment Management, our case environment.

3. Design of the aggregate EPC

Several approaches can be conceived to arrive at the design of an aggregate EPC, such that it allows for creating aggregate process models that refer to different products, services, or customer groups. Therefore, in Section 3.1, we will first specify a set of *design requirements* that we consider relevant for an aggregate EPC. Next, in Section 3.2, we will discuss various alternative designs that fulfill these requirements and motivate our selection from these alternatives. Finally, we will introduce in Section 3.3 a so-called *product hierarchy* as a means to efficiently support the representation of the aggregate EPC.

Note that in the remainder of this paper, we will mostly refer to "product" as the aspect that distinguishes different process models from each other while at the same time being very similar in many other respects. However, one may also think of, for example, different "services" or "customer groups" that require (partly) different processes for their support.

3.1. Requirements

When aggregating process models in a naive, straightforward way, there is a risk of loosing information about the model context. In other words, it may be difficult for parts of the aggregate model to determine to which product it belongs. This is illustrated in Fig. 2.

In this figure, we see at the left-hand side so-called *singular* process models for products *A* and *B*. These can be combined into an aggregate model in a straightforward manner, such as depicted at the right-hand side of the figure.[2] However, on the basis of this aggregate model alone it is not possible to *precisely* distinguish the process flow for *A* from the one for *B*. Suppose that we want to determine the process model for either *A* or *B* on the basis of the aggregate model and we have no other information than the aggregate model itself. Then, it is not clear whether the path from the top *XOR* connector to the bottom *XOR* connector has to be followed for *A* or *B*. Similarly, it is not possible to establish whether event *E2* and function *F2* belong to the process of *A* or to the process of *B* (or perhaps to both). The former observation leads to the following design requirement:

(*R1*) At any time it must be possible to extract from the aggregate model again the original, singular process models. This means that neither superfluous paths or nodes in such extracted models are allowed, nor missing paths or nodes. Clearly, as a first step towards accommodating this requirement it is necessary to include some of the *context* of the original models into the aggregate model. We now also formulate the following design requirement:

(*R2*) The context information from the original models needs a *graphical* manifestation in the process diagram of the aggregate model.
For the modeling experts who are concerned with the maintenance of the aggregate model, the fulfilment of requirement *R2* can be expected to allow for a higher model comprehension [13], which in its turn has a favorable effect on the maintainability of the model. Requirement *R2* is also clearly in the spirit of EPCs, which aim to visualize all important information.

[2] Note that one cannot decide to simply merge events *E2* and *E3* because of the different business semantics these labels may carry.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

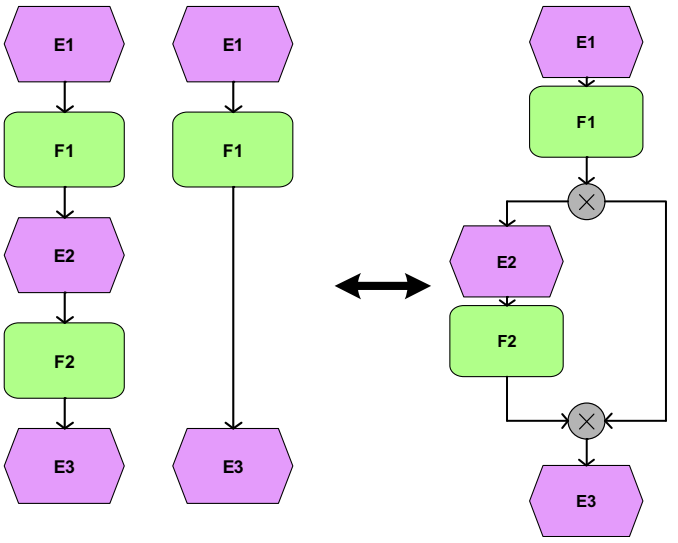

Fig. 2. The process models for A and B are combined into an aggregate model.

In the remaining part of this section, we will focus on requirement *R2*, which deals with the representation of the aggregate model. In Section 4, we will address how to extract the original, singular models from an aggregate model (*R1*): this is a much deeper issue, which deserves more elaboration.

3.2. Representation alternatives

To fulfill requirement *R2*, which states that context information on single process models has to be added graphically to the process model, the use of *labels* in the aggregate model seems a promising approach. We have considered five alternative implementations that build on the use of such labels, which we will review here shortly.

First of all, the use of configurable EPCs was considered. Configurable EPCs (C-EPCs) have been proposed as a modeling language to capture reference models for the configuration of Enterprise Systems, such as SAP [35]. The special feature of C-EPCs is that – unlike most existing reference modeling languages – they allow for the inclusion of configurable elements, i.e. configuration decisions that have to be made in transferring a reference model to an actual implementation. In this way, one configurable EPC allows for the derivation of a set of different model variants, each representing a different configuration of a general model. Note that this one-to-many relationship between models is similar to how one aggregate EPC integrates an arbitrary number of process models for different products. A particular feature of C-EPCs is that connectors can be labeled with logical expressions to specify the configuration options at such a point. Similarly, such logical expressions could be used in an aggregate EPC to label connectors, in this way indicating which of its outgoing (or incoming) paths relate to which particular product. Unfortunately, even though such logical expressions can be conceived and processed by experienced process modelers, they are much too complex to understand by most business users. Our experiences with trying C-EPCs in the business environment of ING Investment Management confirmed this expectation.

As a second alternative, we experimented with moving the complex product logic at the individual connectors to their successor nodes. For example, instead of specifying at an *XOR*-split that the outgoing path that starts with function *A* relates to product *P and* the outgoing path that starts with function *B* relates to product *Q*, function *A* would simply carry a 'product *P*' label and function *B* a 'product *Q*' label. This solution turned out to be unsatisfactory too. Even though the labels themselves could be understood by business users, they found it difficult to understand for unlabeled parts of the aggregate EPC to which products they referred to.

Based on these experiences, it became clear to us that *simple* labels (i.e. labels without any logic) with an *extensive* distribution over the aggregate EPC would be the preferable way to implement requirement *R2* in an actual business domain. Note that in [36] a similar labeling approach is taken, as so-called *ids's* are associated to activities in an integrated process model. That usage, however, takes place within the context of process models that are created primarily for execution purposes – not communication.

We considered three levels of labeling the various nodes in an aggregate model, where a label simply lists the set of products for which this node is relevant:

(1) to label all functions,
(2) to label all functions and events,
(3) to label all functions, events, and connectors.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

As best trade-off, option (2) emerged, an example of which is shown in Fig. 3. In this figure, it can be seen that function *B* is carried out for product *ABA*, while function *C* is carried out for product *ABB*. This leads to the following formal definition of an aggregate EPC.

Definition 6 (Aggregate EPC). An *aggregate EPC* (aEPC) is a septuple (E, F, C, l, A, AL, al), where:

- (E, F, C, l, A) is a syntactically correct EPC (see Definition 5).
- AL is a finite (non-empty) set of labels.
- $al : (E \cup F) \rightarrow (\mathcal{P}(AL)) \setminus \{\emptyset\}$, is a function which maps each function and event onto a non-empty set of labels.

From this definition it follows that an aEPC is an EPC to which labels have been attached to *all* functions and events. Like an EPC, it consists of three types of nodes: events (E), functions (F) and connectors (C). Connectors can be of the type $\vee$, *OR* or *AND*, which is defined by function l. These three types of nodes can be connected by arcs, which is defined by relation A. The labels in an aEPC are represented by the set AL. Function al attaches these labels to the functions and events. It is allowed to attach one or more labels to a function or event.

Even though the use of aEPCs turned out to be an acceptable and understandable way to represent aggregated process models to business users, there is a risk that the process model becomes swamped with product labels in case of many different products. In the next subsection, we will deal with this issue.

3.3. Product hierarchy

When many different products exist also many labels may occur along the nodes in an aEPC. But even though all such products differ from each other in some respect, some products are "more equal than others". In other words, some products together can be considered as a *sub-family* of the entire *family* of products. A good way to limit the number of product labels in an aEPC, therefore, is to exploit the similarities between various products and refer with specific labels to such

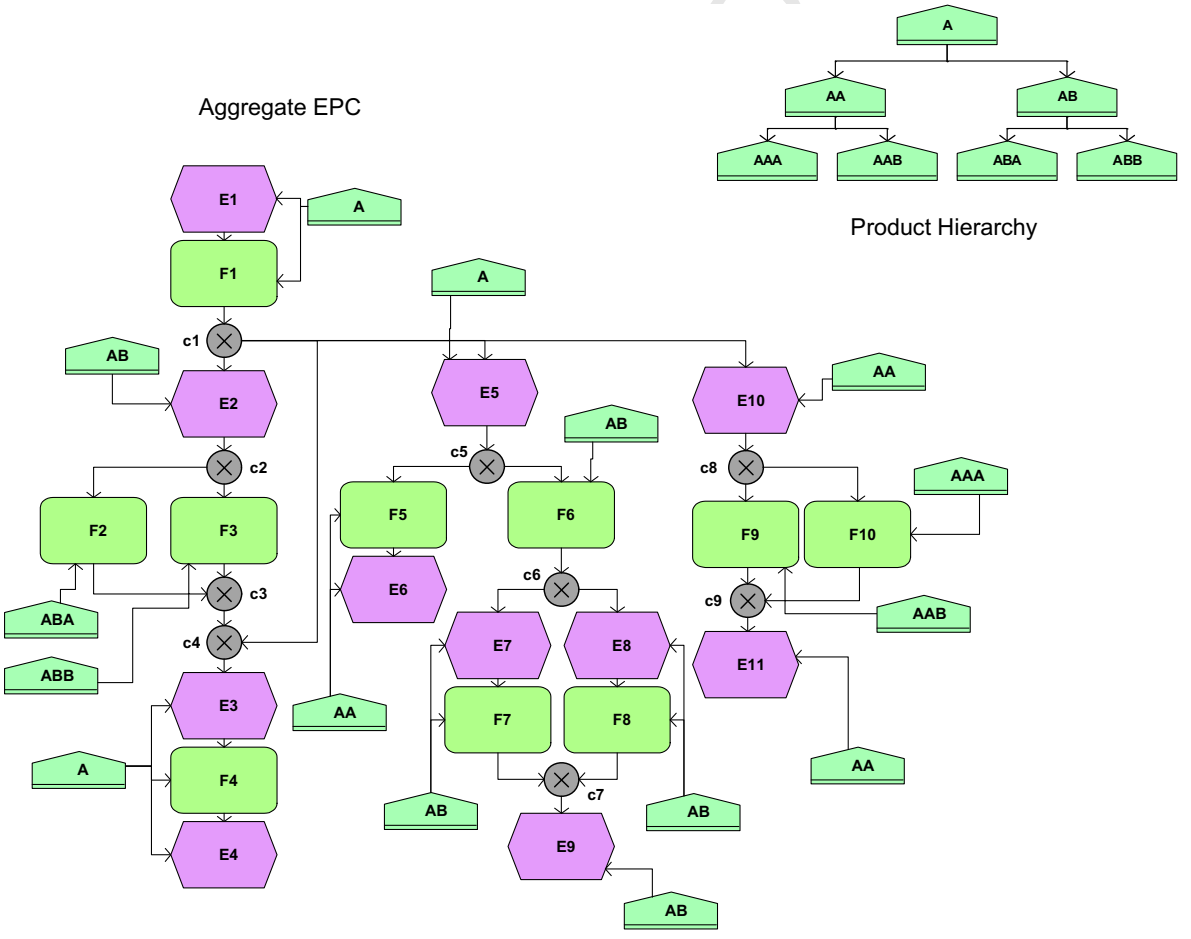

Fig. 3. Process model lead example.

sub-families instead of to all their members. The inspection of product aspects that are of great influence on the actual processing of such products can help to identify such similarities, for example: the variables that are used to make routing decisions in a process model.

To support the idea of labels representing multiple products, we introduce the *product hierarchy*. An example is shown in the top right corner of Fig. 3. In a product hierarchy we distinguish between two kinds of labels: leaf labels and hierarchy labels. Each label that is attached to a leaf in the hierarchy is a *leaf label*; it represents a *single* product. A label that is attached to a node that is not a leaf is a *hierarchy label*; it represents a *set of products*, namely all those products represented by the labels of the leaves to which there are directed paths from that node.[3] In practice, the name of a hierarchy label will often be a *common property* of the set of products it represents. So, the relationship between a node and its children in the product hierarchy can be understood as a *generalization relationship* [9] and the hierarchy as a *specialization/generalization* hierarchy. This can be illustrated by the following example. Let us assume that the leaf labels *AAA* and *AAB* in the product hierarchy of Fig. 3 correspond with mortgage products that respectively have a *variable interest rate* and a *fixed interest rate*. In that case, the hierarchy label *AA* can be used to represent *mortgages* of both kinds, which can be seen from the directed paths that lead from *AA* to both *AAA* and *AAB*.

This leads to the following definition of a product hierarchy.

Definition 7 (Product hierarchy, aEPC match). A *product hierarchy* is a rooted tree $RT = (V, D)$. It is said to *match* with an aEPC (E, F, C, l, A, AL, al) when $AL \subseteq V$.

A product hierarchy that matches a particular aEPC expresses for each of the labels in the aEPC (1) whether it represents a single product or multiple products and (2) to which product(s) it refers.

By introducing a product hierarchy, the number of labels in an aEPC can be reduced in comparison to the situation where products have to be identified with an individual label. This reduction is likely to positively influence both the understandability and maintainability of the model. After all, process models with fewer elements are generally easier to understand [27] and contain fewer errors than larger models [28].

Even though in the context of this paper we will use exactly one product hierarchy that matches a given aEPC, it can be imagined for a business scenario that multiple product hierarchies are in use simultaneously. In this way, orthogonal categorizations of the various functions and events can be given. For example, in the setting of ING Investment Management separate product hierarchies are maintained for (1) product groups and (2) customer groups.

Note that the use of a product hierarchy does not prohibit associating *a set* of labels to a separate event or function, as can be seen from Definition 6. Furthermore, *if* a set of labels is associated to an event or function, it may consist of a list of leaf labels, hierarchy labels, or a mix of both. It is up to the modeler to make the trade-off between, for example, introducing a label for a new sub-family or to use a listing of existing labels to characterize an event or function.

4. Extracting singular EPCs from an aggregate EPC

This entire section is devoted to requirement *R1*, which was identified in Section 3.1 but has not been satisfied yet: at any time it must be possible to again extract from an aggregate process model the original, singular process models. Although aEPCs are beneficial to reduce the amount of similar process models, process models for specific products are still needed by various stakeholders at various occasions. For example, consider a manager who wants to provide work instructions to a new employee. If the employee will be concerned with a limited set of products, such an instruction preferably takes place on the basis of a process model that is specific for this sub-family of products.

To illustrate this idea, consider the aEPC in Fig. 3. The model describes the aggregated process for the products *AAA*, *AAB*, *ABA* and *ABB*. The labels indicate that a function or event belongs to the process of one or more of these products.[4] Let us assume that a manager wants to show the process for products *ABA* and *ABB*. As can be seen in the product hierarchy at the top right corner of Fig. 3, these products together form a sub-family of products that is labeled *AB*. Clearly, the aEPC in Fig. 3 also contains nodes that are of no relevance for the new employee concerned with the *AB* process, like the function *F10*, which is labeled with *AAA*, and event *E10*, which is labeled with *AA*. Neither *AAA* nor *AA* are part of the *AB* sub-family.

It would be time-consuming and error-prone to manually remove the irrelevant nodes from an aEPC, restore the logic flows between them, and generate again an aEPC for a specific product or set of products. An automatic solution is therefore preferable, provided that it exactly generates the functions, events, connectors, and paths that are relevant for the execution of the process of the products under consideration. In this section, we will explore what such an algorithm should look like.

[3] Note that we allow for hierarchy labels that represent a single product only. This is particularly useful for product hierarchies that are under construction.

[4] Note that the connectors in Fig. 3 carry labels too. They are no part of the formal definition of an aEPC (see Definition 6) and are only added here to clarify the manipulation of this model in the remainder of this paper.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

For the remainder of this section, we will conduct a "thought experiment" for the aEPC in Fig. 3. We will assume an interest in the particular *AB* sub-family of products and informally describe the steps to extract a process model from this aEPC. Along the way, requirements and issues will be addressed.

4.1. Selecting functions and events

If one would carry out our thought experiment, an obvious question would be: which nodes in Fig. 3 are relevant in the context of *AB*? Clearly, at least the functions and events to which label *AB* itself is directly attached seem relevant, i.e. the events *E2*, *E7*, *E8* and *E9* and the functions *F6*, *F7* and *F8*.

Additionally, it can be seen in the product hierarchy that *A* is a *predecessor* of *AB*, i.e. *A* is a *generalization* of *AB*. So, if it is specified that a task is relevant for the execution of *A*, this means that it is relevant for the execution of *AB* as well. The same holds for the events: when some event belongs to the *A* process, they also belong to the *AB* process. So, in our case of extracting the process for *AB*, we also need to select the events *E1*, *E3*, *E4* and *E5* and the functions *F1* and *F4*, because label *A* has been attached to those. Note that if *A* was not the root and would have had a predecessor of its own, then that node would need to be considered too.

In the product hierarchy, it can also be observed that *AB* has *ABA* and *ABB* as *successors*, i.e. *ABA* and *ABB* are *specializations* of *AB*. In fact, *ABA* and *ABB* are the leaf labels representing the actual products that the *AB* family consists of. What does this mean for the intended process model that we wish to extract? Clearly, when a task is executed for either *ABA* or *ABB* it concerns the products that are part of the *AB* family. The most desirable guideline that comes forward from our practical use of aEPCs and product hierarchies within ING Investment Management is that somebody who is interested in a particular product family will also be interested to see the tasks that are executed for *each* of the more specific products it involves. And, of course, this holds for events as well. For our example, this means that we also need to select the functions *F2* and *F3*, because labels *ABA* and *ABB* have been attached to them, respectively. Note that this interest would have extended to further successors as well, would they have existed.

So, in general, when extracting a process model for a particular product, all functions and events that are labeled with either generalized or specialized forms of that product should be selected. To select those functions and events we introduce a new function, which builds on the notion of root paths we introduced earlier (see Definition 2).

Definition 8 (Function H). For an aEPC (E, F, C, l, A, AL, al) with matching product hierarchy $RT = (V, D)$ and $x \in V$, H is defined as follows: $H(RT, x) = \{y \in V | \langle v_1, v_2, \ldots, v_n \rangle$ is a root path $\wedge\, x = v_i$ for some $i, 1 \leqslant i \leqslant n \wedge y = v_j$ for some $j, 1 \leqslant j \leqslant n\}$.

The functions and events that need to be selected to extract a model for a product that is represented by a specific label in a product hierarchy can now be determined as follows: it is the union of all products represented by the nodes in the product hierarchy that are on the root paths through that node, which can be determined with H.

4.2. Selecting connectors

Until now, we did not say anything about which *connectors* to select when extracting a singular product model from an aEPC. It follows from our choice not to label connectors (see Definition 6) that it is not directly obvious which connectors relate to which products. It is important in this respect to distinguish three different situations for connectors. After all relevant functions and events are selected in an aEPC there will be:

(1) connectors through which *no* path leads from a node that is selected to another node that is selected (e.g. *c8*),
(2) connectors for which there is a *single* path that leads from a node that is selected to another node that is selected (e.g. *c5* is on a path from *E5* to *F6*), and
(3) connectors which are on *multiple* paths between nodes that are selected (e.g. *c6* is on a path from *F6* to *E7* and on a path from *F6* to *E8*).

For the nodes in the first category, it is quite obvious that such connectors should not appear in the extracted model as they do not join any of the selected nodes in the aEPC. Also, connectors in the second category, which have just a single input and single output, are superfluous in determining the possible flows: only a single path through them can be followed. Such connectors are not allowed in EPC's (see Definition 5). For this category of connectors, it is sufficient to preserve the path but *without* the connector that was originally on it. Finally, the third category of connectors is unproblematic: such connectors are on multiple paths and should appear in full in the extracted model. If we would continue our thought experiment in this way, we arrive at the model that is shown in Fig. 4.

As can be easily verified, the resulting model is a syntactically correct EPC. For the example connectors we just mentioned, we can see that:

(1) *c8* does not appear in the model,
(2) *c5* is also removed but the path between *E5* and *F6* is still present, and
(3) *c6* is preserved including both paths that run through it from *F6*, i.e. to *E7* and *E8*.

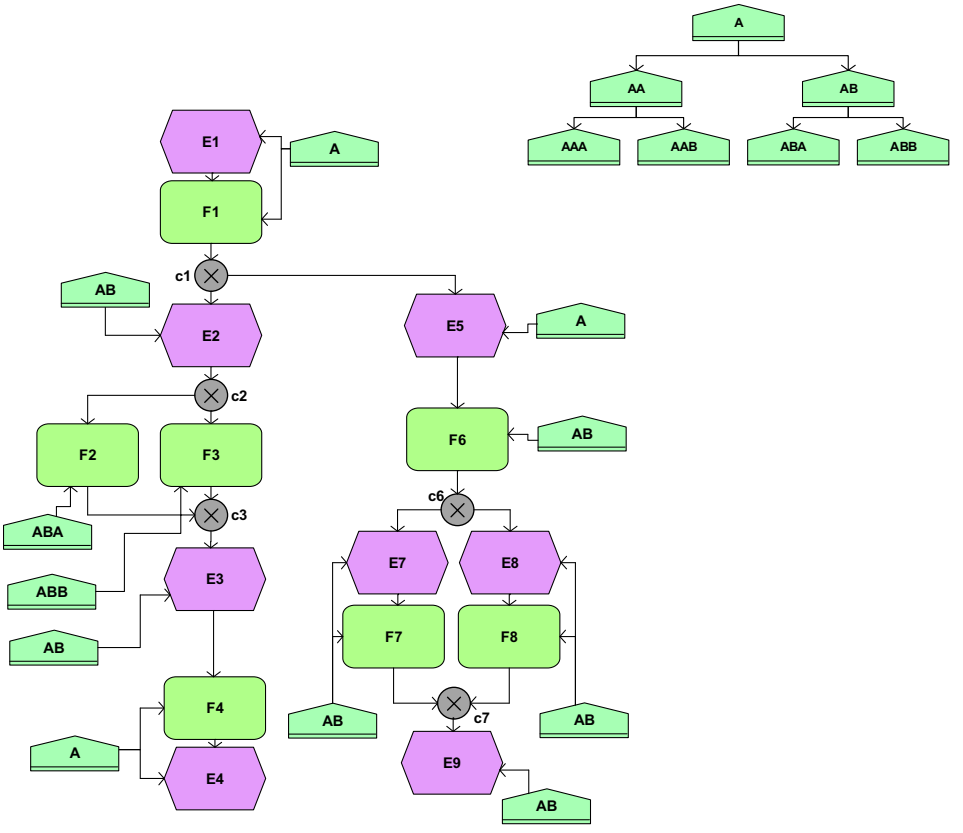

Fig. 4. Result lead example after extraction for *AB*.

In many practical cases, this approach works satisfactorily. However, a particular style of creating aEPCs may lead to ambiguous results, as will be elaborated in the following section.

4.3. Solving ambiguities

Consider the simple aEPC and product hierarchy as shown in Fig. 5. The latter consists solely of root *A* and products *AA* and *AB*. Suppose now that we would be interested in product *AA* and want to extract a process model from the aEPC in the fashion as described so far. We would then arrive at the process model in Fig. 6.

In the latter figure, we see that all events and functions are preserved from the aEPC that either carried label *A* (e.g. event *E1*) or label *AA* (e.g. event *E2*). Event *E4* has been removed because it is labeled with *AB*, which is neither a successor nor a predecessor of *AA* in the product hierarchy. Also, connectors are preserved that support more than a single path between selected nodes. For example, *c2* lies on a path from function *F1* to event *E2* and also on a path from function *F1* to event *E3*. But connector *c3* is not included in the model because there is only a single path that runs through it: it leads from function *F1* and connector *c1* to connector *c4* and event *E3*. Therefore, *c3* is removed and only the path between *c1* and *c4* is preserved.

What is curious here is that the extracted model offers a path for product *AA* that does not include event *E2*, i.e. in case *E3* directly occurs after the execution of function *F1*. In the original aEPC, this is a situation that the modeler *may* have wanted to avoid by using the *XOR-split* connector *c1* after function *F1* (see Fig. 5). This connector could then be interpreted in this model as the implicit choice between carrying out the process for either product *AA* or product *AB*. In the former case, both events *E2* and *E3* would take place; in the latter case, there is a further choice between which of the events *E3* or *E4* will happen.

The problem is that we do not exactly know what the modeler intended, the original model is ambiguous. Part of the ambiguity problem here is caused by our choice to not label connectors, because of the overload of connectors in an aEPC this will generate (see Section 3.2). Another part of this ambiguity is caused by the modeling style that is applied here.[5] To address this issue we will distinguish between two different types of connectors, using a new function *edge*. Using this distinction we can identify ambiguous situations before we commence to extract a singular process model from an aEPC.

[5] Note that even though our example involves the ambiguous use of a *split* connector, a similar example can be constructed that uses a *join* connector.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

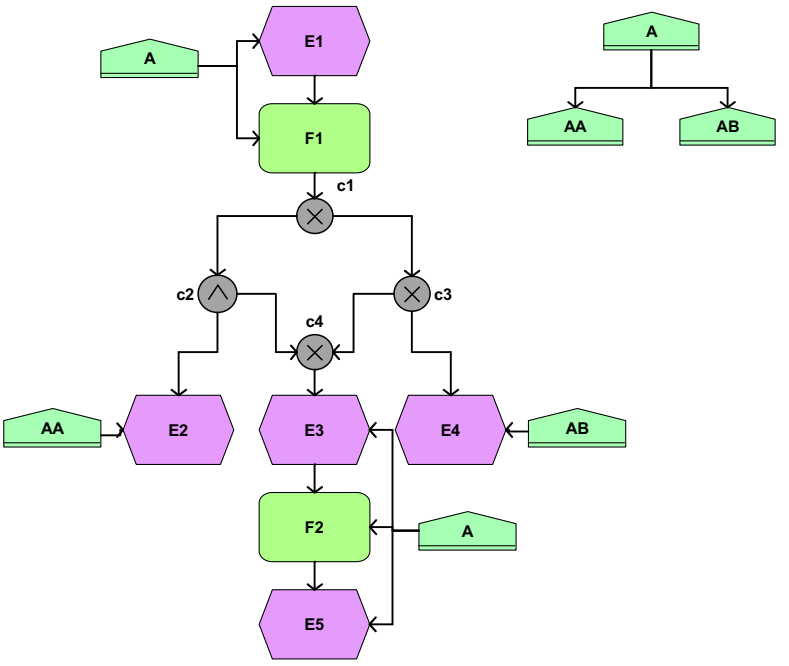

Fig. 5. Simple process model.

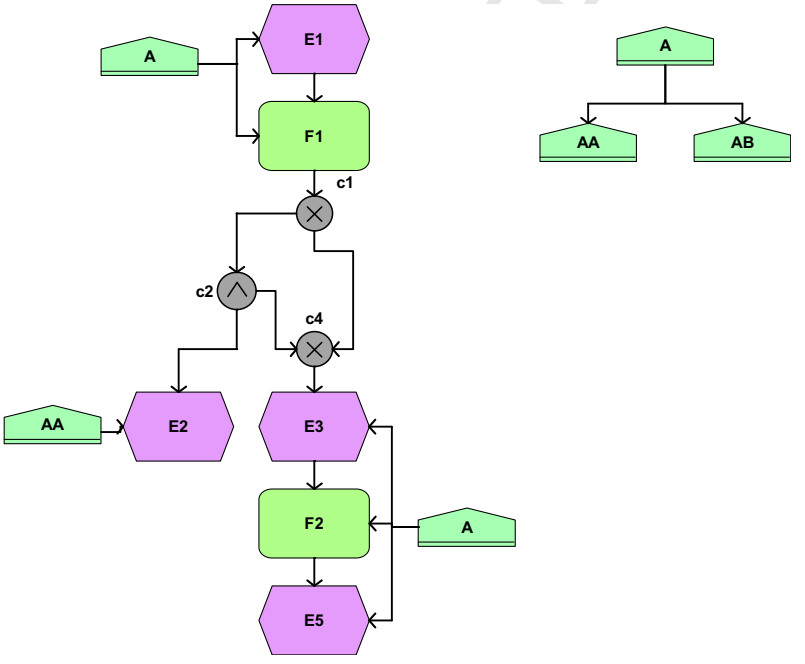

Fig. 6. Result of *AA* extraction from the simple process model.

Definition 9 (Edge function). For an aEPC (E, F, C, l, A, AL, al) with matching product hierarchy $RT = (V, D)$, its associated function $edge : A \to \mathscr{P}(AL)$ is defined as follows:

For any $(a_1, a_2) \in A, x \in edge(a_1, a_2)$ if and only iff:

- $x \in leaves(RT)$, and
- there is a path $\langle v_1, v_2, \ldots, v_n \rangle$ in the aEPC for some $n \geqslant 2$ such that

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

- $v_1, v_n \in (E \cup F)$,
- $v_i \in C$ for $1 < i < n$,
- $(\exists y, z : y \in al(v_1), z \in al(v_n) : x \in H(RT, y) \cap H(RT, z))$,
- $a_1 = v_j \wedge a_2 = v_{j+1}$ for some $1 \leqslant j < n$.

With the definition of the *edge* function, for each arc in an aEPC it can be determined *for which products it is used*. This is determined by first considering all paths, flowing from a single event or function to an event or function immediately following it, that such an arc is on. Note that in an EPC, multiple connectors may succeed each other in between events and functions. Secondly, all products are considered that relate to both the start and the end node of such a path, where a node can either be a function or event. Only *single* products are considered here, i.e. the products that are represented as leaves in the product hierarchy. If a product is present as label of both the start and end node of such a path, the arc in question is considered to be *in use* for that product. If a more generalized form of that product is present as label at both the start and the end node of this path, this is the case too. Note how we can check this using H (see Definition 8).

The particular application of the *edge* function useful to us now is to determine for which products the incoming and outgoing arcs of connectors are used. By doing so, we can distinguish between product-dependent and product-independent connectors, as follows.

Definition 10 (C_{indep}, C_{dep}). For an aEPC (E, F, C, l, A, AL, al) with matching product hierarchy (V, D), the sets of product-independent connectors $C_{indep} \subseteq C$ and product-dependent connectors $C_{dep} \subseteq C$ are defined as follows:

- $c \in C_{indep} \iff (\forall x, y : x \in \bullet c, y \in c\bullet : edge(x, c) = edge(c, y))$,
- $C_{dep} = C / C_{indep}$.

For a *product-dependent connector*, at least one of its incoming arcs is used for a set of products that is different than is the case for one of its outgoing arcs. For a *product-independent connector*, the incoming and outgoing paths are used for exactly the same set of products. Intuitively, we distinguish here between connectors that respectively *do* take the difference of products into account and those that do not.

In Fig. 7, we can see the original model again from Fig. 5 where *for illustration purposes only* along all arcs the single products are listed for which they are used, on the basis of function *edge*. The edge labels are no part of an aEPC (see Definition 6), and are normally not shown to end-users. From the figure, it can be seen that connectors *c1* and *c4* are product-independent connectors, while *c2* and *c3* are product-dependent connectors. Intuitively, this means that for *c1* and *c4* the exact product type is not relevant for the split behavior of these connectors. For *c2* and *c3*, different logical flows for different subsets of products can be distinguished. In other words, their behavior depends on a distinction between products.

A non-ambiguous alternative to the example process would be the model as shown in Fig. 8. In contrast to the earlier model, event *E3a* is inserted between *c2* and *c4* and event *E3b* is inserted between *c3* and *c4*; they are labeled with *AA*

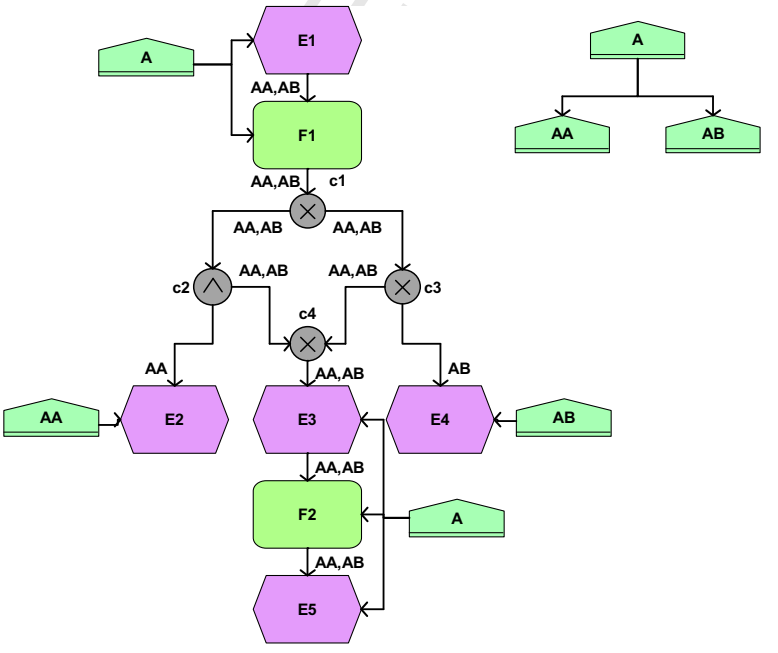

Fig. 7. Simple process model with the result of function *edge* next to each arc.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

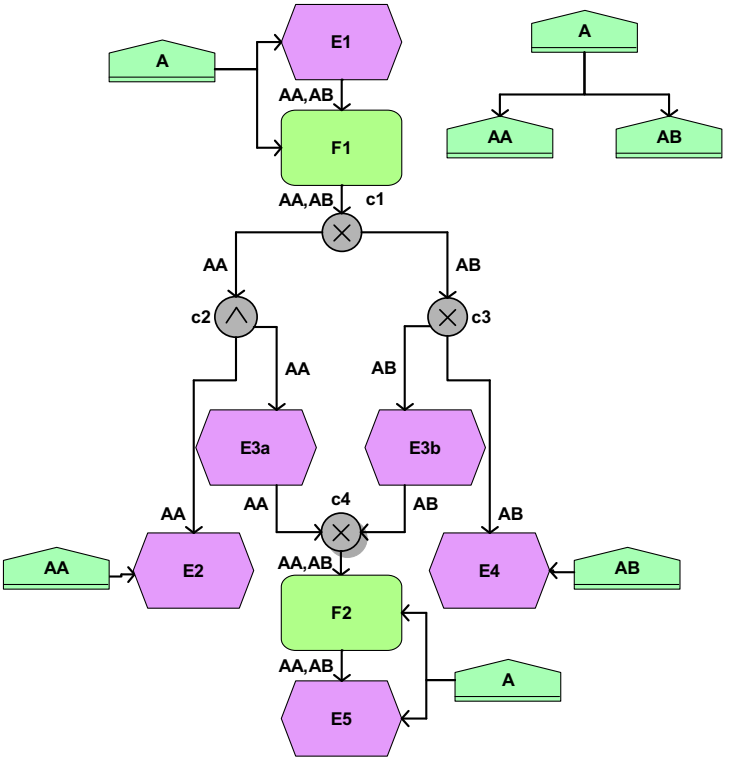

Fig. 8. Non-ambiguous alternative to the simple process model.

and *AB*, respectively. In this way, our interest in product *AA* would lead to the selection of event *E1*, function *F1*, event *E2*, event *E3a*, function *F2*, and event *E5*. By selecting the connectors as elaborated in the previous section, this will lead to the process model as can be seen in Fig. 9. Note that in this model it is impossible to skip event *E2*, which was the problem for the example model of Fig. 5.

Ambiguity problems like the one in the example we discussed can be effectively prevented by prohibiting the extraction of singular process models from an aEPC if the latter contains:

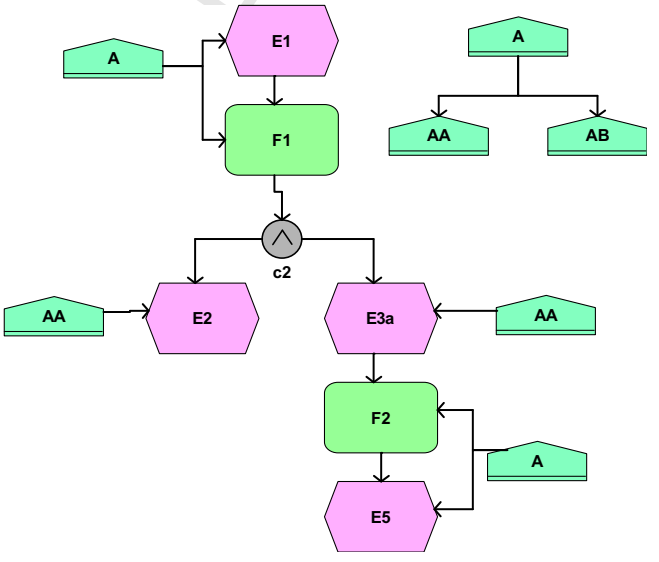

Fig. 9. Result of *AA* extraction from the alternative process model.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

(1) A product-dependent *split* connector that is not directly *followed* by functions or events for its outgoing arcs.
(2) A product-dependent *join* connector that is not directly *preceded* by functions or events for its incoming arcs.

By preventing these situations, we basically only allow for sequences of connectors if their combined, logical behavior is *independent* of product characteristics. Otherwise, we require *labeled nodes* to be interleaved with such connectors – thus removing ambiguity problems. This is exactly the way that the non-ambiguous alternative in Fig. 8 was created, i.e. by inserting events *E3a* and *E3b* after the product-dependent connectors *c2* and *c3*. Note that the prevention measures are sufficient but not necessary conditions to avoid ambiguity problems. For example, the non-ambiguous alternative to the simple process model still contains product-dependent connector *c1* that is not directly followed by a function or event. It is easy to see how the above conditions can be met by including more events and functions into this model.

4.4. Correctness

By now, it is clear what elements need to be selected from an aEPC to extract a singular process model of interest. If we consider the singular process model that results from such an extraction, it seems desirable that its events and functions also carry the original labels of the aEPC it has been extracted from. As we explained, nodes may have been selected that are carried out for specialized or generalized forms of the product or product sub-family of interest. Stakeholders will want to have access to this information to distinguish the nodes. Therefore, it is desirable that the singular process model is once more an aEPC, i.e. an EPC that is extended with labels from a product hierarchy.

However, for an extracted model to be considered as a correct aEPC it must be so that the underlying EPC should be syntactically correct (see Definition 5). This is not trivial as will be illustrated with the example in Fig. 10. Depicted is a simple aEPC with its accompanying product hierarchy. It is clear to see that it is syntactically correct, conform the requirements in Definition 5.

If we would extract from this aEPC the singular process model for product *AA* in correspondence with the steps we discussed, this would result in a process model that starts with function *F2* and ends with event *E3*. Note that this is not a syntactically correct EPC because (1) it does not contain a start event and (2) it contains a function without an incoming arc from a preceding node, i.e. *F2*. Clearly, if we assume that the procedure is described correctly then this results from the odd labeling style that was used for this example. The following notion of correctness is introduced to mend this problem.

Definition 11 (Correctly labeled aEPC). An aEPC (E, F, C, l, A, AL, al) with matching product hierarchy $RT = (V, D)$ is *labeled correctly* iff:

- $(\forall x : x \in V : (\exists e, y : e \in E \wedge y \in AL : |\bullet e| = 0 \wedge y \in al(e) \wedge x \in H(y)))$,
- $(\forall x : x \in V : (\exists e, y : e \in E \wedge y \in AL : |e \bullet| = 0 \wedge y \in al(e) \wedge x \in H(y)))$,
- $(\forall f, a_1, a_2 : f \in F \wedge (a_1, f) \in A \wedge (f, a_2) \in A : edge(a_1, f) = edge(f, a_2) = (\bigcup y : y \in al(f) : H(y)) \cap leaves(RT))$.

In this definition, the first requirement states that for each product in the product hierarchy a start event exists such that one of its labels is on the same root path as that product in the product hierarchy. The second requirement states that also an end event exists with this property. In this way, it is guaranteed that – regardless of the product under consideration – always at least one start and at least one end event exist in an extracted aEPC for any product or product sub-family.[6] Note that the aEPC in the example violates both the first and the second requirement of Definition 11.

The third requirement deserves some special attention. In the first place, it states that for each function its incoming path is used for exactly the same set of products as its outgoing path, which can be determined with *edge* from Definition 9. Recall that *edge* generates labels for leaf nodes only, i.e. it refers to single products. Furthermore, it is required that this set is exactly the same as the set of single products that is associated with a particular function. The latter set can be determined by using *H* on all the labels of this function and intersecting the joined result with the leaf labels in the product hierarchy. Because in our example, the following holds:

- $edge(E2, F2) = \{AB\}$,
- $edge(F2, E3) = \{AA\}$, and
- $(\bigcup y : y \in al(F2) : H(y)) \cap leaves(RT) = H(A) \cap leaves(RT) = \{AA, AB\}$.

It is clear that this third requirement is violated as well. The incoming and outgoing arcs for function *F2* are somehow in use for different products – $\{AB\}$ versus $\{AA\}$ – which hints at an undesirable discontinuity. What is more, the label *AA* of function *F2* points to yet another set of products, i.e. $\{AA, AB\}$. As will be shown in the next section, these requirements will play a role in the formal proof of the claim that the proposed extraction approach – provided that the original aEPC is correctly labeled – once more generates an aEPC.

[6] A pragmatic way used within ING Investment Management to meet these requirements is to have at least one start and one start event labeled with the root element of the product hierarchy.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

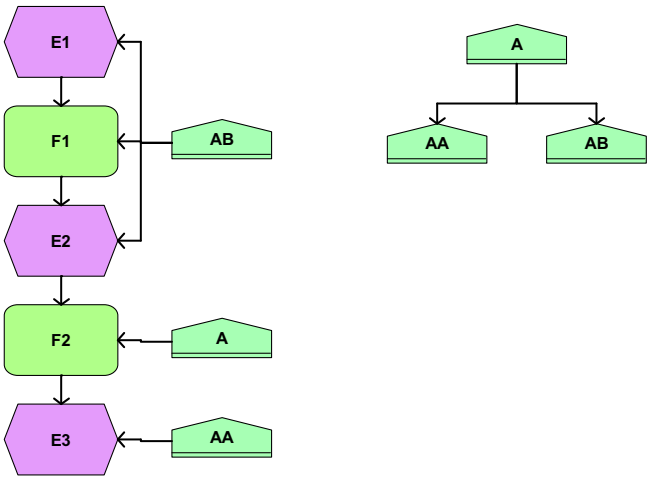

Fig. 10. Incorrectly labeled aEPC.

This ends our exploration of extracting process models from aEPCs. It has led us to a blueprint for the formal specification of the extraction algorithm, a way of preventing an ambiguous modeling style, and criteria to decide on the correct labeling of aEPCs.

5. Extraction algorithm

Based on the informal exploration in the previous section of how to extract a singular process model from an aEPC, the algorithm for this purpose is presented in pseudo-code as shown below.

Algorithm 1 (Extraction of a singular model from an aggregate EPC). Let P be an aEPC (E, F, C, l, A, AL, al) that is correctly labeled in correspondence with the matching product hierarchy $RT = (V, D)$. If we consider product label $pl \in V$, *extract* is the function that generates a product-specific aEPC Q for product label pl from V. Hence, $Q = extract(P, pl)$.

Q is constructed in a number of steps, as follows:

(1) IF $(\exists c_1, c_2 : (c_1, c_2) \in A : c_1 \in C_S \cap C_{dep} \vee c_2 \in C_J \cap C_{dep})$ THEN {generate warning; END}

(2) $Q := (E', F', C', l', A', AL', al')$ with
- $E' = \{e \in E | \exists x : x \in H(RT, pl) : x \in al(e)\}$
- $F' = \{f \in F | \exists x : x \in H(RT, pl) : x \in al(f)\}$
- $C' = \{c \in C | \langle v_1, v_2, \ldots, v_n \rangle$ is a path in P for some $n \geqslant 3$ such that
  - $v_1, v_n \in (E' \cup F')$,
  - $v_2, \ldots, v_{n-1} \in C$, and
  - $c = v_i$ for some $2 \leqslant i < n\}$
- $l' = l|_{C'}$
- $A' = A \cap ((E' \times F') \cup (F' \times E') \cup (E' \times C') \cup (C' \times E') \cup (F' \times C') \cup (C' \times F') \cup (C' \times C'))$
- $AL' = AL \cap H(RT, pl)$
- $al' = al|_{(E' \cap F')}$ is a model in which we have the functions and events that correspond to product label pl and the relevant connectors on the paths between those. There may, however, be connectors in this model that have one incoming and outgoing arc.

(3) $C^* := \{c \in C' | |\{n_1 \in E' \cup F' \cup C' | (n_1, c) \in A'\}| = |\{n_2 \in E' \cup F' \cup C' | (c, n_2) \in A'\}| = 1\}$ is the set of connectors with one incoming and one outgoing arc, taking into account the dependencies within Q.

(4) WHILE $C^* \neq \emptyset$ DO{
- select an arbitrary $c \in C^*$, with
  - $(n_1, c) \in A'$ for some $n_1 \in E' \cup F' \cup C'$, and
  - $(c, n_2) \in A'$ for some $n_2 \in E' \cup F' \cup C'$[7]
- $C' := C' \setminus \{c\}$
- $A' := (A' \cup \{(n_1, n_2)\}) \setminus \{(n_1, c), (c, n_2)\}$
- $C^* := \{c \in C' | |\{n_1 \in E' \cup F' \cup C' | (n_1, c) \in A'\}| = |\{n_2 \in E' \cup F' \cup C' | (c, n_2) \in A'\}| = 1\}\}$;

END

[7] Note that n_1 and n_2 are unique because c is taken from C^*.

In step (1), the algorithm checks whether there are product-dependent split (join) connectors that are not preceded (followed) or followed by (labeled) events or functions. If so, the algorithm halts because the aEPC is ambiguous. Otherwise, the algorithm continues.

In step (2), the algorithm prunes irrelevant events, functions, and connectors from the original aEPC. Only those events and functions that are either directly labeled with the product of interest or with a product that is either a specialized or generalized form of that product are retained. Also, all the paths and connectors between retained nodes are retained themselves.

In step (3), the set of connectors is determined that have a single incoming and a single outgoing arc. These connectors are superfluous and not allowed in an EPC.

In step (4), there is a check on whether there are (still) connectors left with a single incoming and a single outgoing arc. If there are, one of these is selected, its predecessor and successor node are directly connected, and the connector itself is removed. This process continues until there are no such connectors left.

Following these steps, all relevant nodes and the paths through them are selected. Due to the initial ambiguity check, no superfluous paths are included in the model. What remains to be proven is that application of the algorithm results in an aEPC itself.

Theorem 12 (Extraction result is an aEPC). *Let P be an aEPC (E, F, C, l, A, AL, al) that is correctly labeled in correspondence with the matching product hierarchy $RT = (V, D)$. If $Q = extract(P, pl)$ for some $pl \in V$, then Q is an aEPC.*

Proof. Let $Q = (E', F', C', l', A', AL', al')$ be the result of $extract(P, pl)$. What needs to be proven is given in Definition 6. The only part from it for which it is not immediately obvious that it is satisfied, is whether (E', F', C', l', A') is a *syntactically correct* EPC. We will consider the requirements one by one (see Definition 5):

- It is easy to see that sets E', F', and C' are pairwise disjoint, because E, F, and C are so too.
- Potentially, only the addition of arcs can violate the requirement that all events in E' have at most one incoming and one outgoing arc. The only place where this can take place in algorithm *extract* is step (4), where an event may be directly linked to another node when a superfluous connector is removed. However, the arc to that superfluous connector is then removed as well. So, both the number of incoming and outgoing arcs for each event $e \in E'$ is the same as it was in E.
- There is still at least one start in Q, because P is correctly labeled (see Definition 11). This means that within P there is at least one start event that carries a label such that it will be selected in step (2).
- For the same reason, there is still at least one end event in Q.
- Because P is correctly labeled, all functions in F' will still have exactly one incoming and one outgoing arc. The reason for this is that the labeling in P is such that for each function $f \in F$ its incoming path is used for exactly the same set of products as its outgoing path, which also equals the set of single products that the function is relevant for (see Definition 11). So, if f is selected to be included in F' on the basis of pl, this is because pl or one of its generalized or specialized forms is part of the labels of f, see step (2) of the algorithm. Let us call this particular label l. But then, there is both a preceding node and a succeeding node of f in P for which this is also the case, i.e. l is a label of those. Otherwise, l could not have been in use for both the incoming and outgoing arc of f (see Definition 9). Both of these nodes will be selected in Q and therefore both the incoming and outgoing arc of f leading from and to these nodes, respectively will be preserved in Q.
- C'_j and C'_s will partition C', just like C_j and C_s partitioned C. After all, the number of incoming and outgoing arcs of connectors that are preserved can only decrease, see step (2).
- C'_{EE} and C'_{FF} will be empty. First of all, no new connectors are introduced in *extract*, so there will be no new elements tying events or functions directly together. Secondly, an already existing connector can only be imagined to have a predecessor function (event) and successor function (event) in Q if that connector would have had a predecessor function (event) and successor function (event) in P. And this is impossible, since P was syntactically correct.
- C'_{EF} and C'_{FE} will partition C', as C'_{EE} and C'_{FF} will stay empty. □

This ends the formal part of this paper. In the next section we will describe how aEPCs support model management in practice.

6. Application of aggregate EPCs

In the previous sections, we presented a method for describing several processes with one aEPC and an algorithm that extracts from such an aEPC a process model that describes the process for a specific product or product group. Furthermore, we explained that the design of the aEPC was guided by our desire to help decrease the number of process models that need to be maintained with a "richer" process model, which is nonetheless still comprehensible for end-users. Consequently, there is a need to show the applicability of our approach in practice. To this end, we will consider as case study the *corporate actions process* from the business environment of ING Investment Management. First we will introduce the context of this company, describe the case study, and then present the results from a wider evaluation of the use of aEPCs within this specific business context.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

6.1. ING Investment Management

ING Investment Management (ING IM) is one of the world's largest asset management companies. The company provides a comprehensive range of investment solutions and services to its clients and partners. The company manages assets for institutional clients, fund distributors, and the labels used within the wider ING Group. ING IM has nearly €400 billion assets under management and a staff of 2500 professionals. It operates along regional lines with centers of expertise in Europe, the America's and the Asia–Pacific region.

In Europe about 100 business processes are in place which produce all the solutions and services to ING IM's clients. All these 100 processes are described in process models with the ARIS Toolset. When these EPCs with their descriptions are printed on A4 paper format, this results in about 500 pages of documentation. About seven people within ING IM's Information and Process Engineering department have a full time job to advice stakeholders in all stages of the Business Process Management Life Cycle. In all analysis, redesign, kaizen, and other process management activities the process descriptions play an important role. However, not only performance improvement drives Business Process Management within ING IM, also compliance and risk management are drivers. *Compliance* means that the processes should comply to laws and regulations. The regulator requires that all relevant processes are described and that these descriptions are indeed compliant. Regulators have the right to request the descriptions at any time and inspect these. Clearly, this implies that there is a large responsibility for the process modelers to keep the descriptions up-to-date. From a *risk management* perspective the process descriptions also play an important role. Attached to the processes are identified risks and controls to mitigate these risks. Based on the descriptions controls are tested. In-control-statements from auditors and accountants are typically based on these test reports. This underlines, also from the Risk Management perspective, that accurate process descriptions are of the utmost importance.

Having stated on the one hand the importance of process descriptions and their actuality, but on the other hand their maintenance burden, this clearly explains the desire for improved model management at ING IM. The main goal is to store process models as efficiently as possible: namely, only once. Clearly, such a situation has many advantages. Updates have to be carried out only on one model instead of on a set of related models. And consequently, consistency of models is less of an issue. Also, a better overview of how models are related emerges when modeled in such an integrated way.

Since products and services within ING IM are often built on a similar set of capabilities, there is also potentially a large business case to create aggregated process models. Services to the client are often a composition of various more basic services which are internal to the organization. And products are built on a standardized set of investment capabilities. Translated to the processes which produce these products and services this implies that large parts of the process models overlap (when aggregate EPCs are not used).

6.2. The corporate action process

The specific process we consider here for illustrative purposes deals with corporate actions. A *corporate action* is defined as an action taken by a public company that has a direct effect on the holdings of its shareholders. As a result of a corporate action, ING IM in its capacity as asset manager must carry out various steps. In total, 15 different services can be classified as a corporate action, each of which requires a slightly different processing by ING IM. These corporate actions can be categorized into three sub categories (or sub-families of corporate actions), i.e. corporate actions:

(1) for which *only cash* can be delivered,
(2) for which *only stocks* can be delivered,
(3) for which a choice between *stock and cash* has to be made.

Basically, we can distinguish here a product hierarchy for corporate actions (root), 3 sub-families (nodes that are not leaves), and 15 single products (leaves).

If an corporate action is taken, then the processing of it by ING IM is roughly as follows. For a corporate action where a choice must be made between cash or stock, the involved investment manager decides on what has to be delivered. After that, a number of checks need to be done in cooperation with some relevant parties, it has to be checked whether everything is processed correctly in the systems of ING IM, and some administrative operations have to be carried out as well. For the last two processing parts, a clear distinction can be made between the activities that have to be done when cash is delivered or when stock is delivered.

At ING IM, the ARIS Toolset is used for modeling the various business processes. The application of our idea to aggregate the various process models into one implied that it must be possible (1) to model aEPCs in the ARIS Toolset and (2) that extractions on aEPCs could be applied automatically. To allow for aEPCs to be created, we were able to simply use the facilities in the EPC diagram notation from the ARIS Toolset. Specifically, this tool allows for building label hierarchies and attaching labels to functions and events. Furthermore, to allow for extracting singular models from the aEPCs, we developed a tool that directly interacts with the ARIS Toolset.

This interaction is facilitated by the general *XML Export/Import* facility of the ARIS Toolset. After that an export to XML has been made from the aEPC and its corresponding product hierarchy, the extraction of a singular process model is executed with our tool and the XML file is *adapted* in such a way that the data in the XML file corresponds with the result of the singular process model. The adapted XML file can then be imported again in the ARIS Toolset and a new diagram, which con-

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

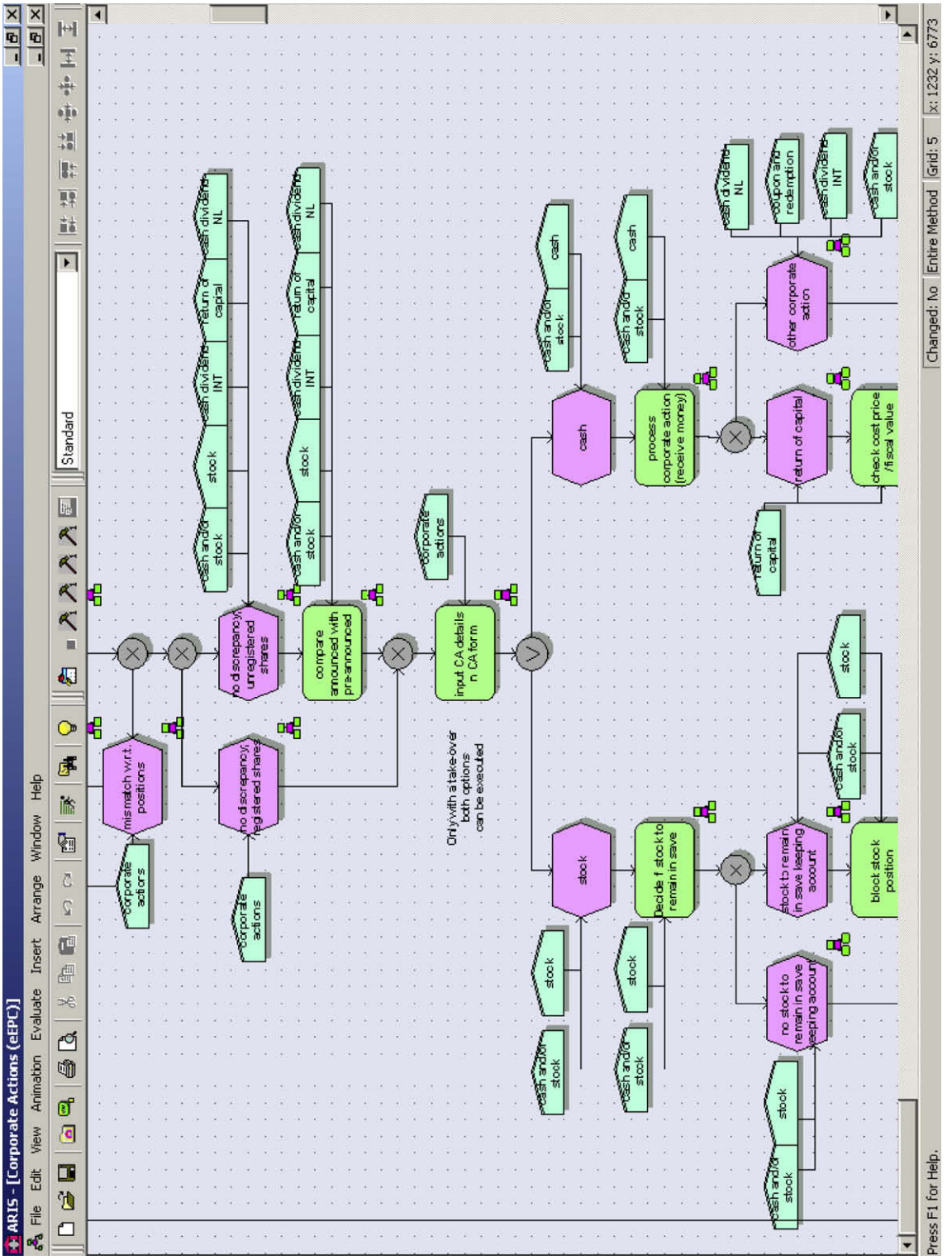

Fig. 11. Part of the aggregate EPC.

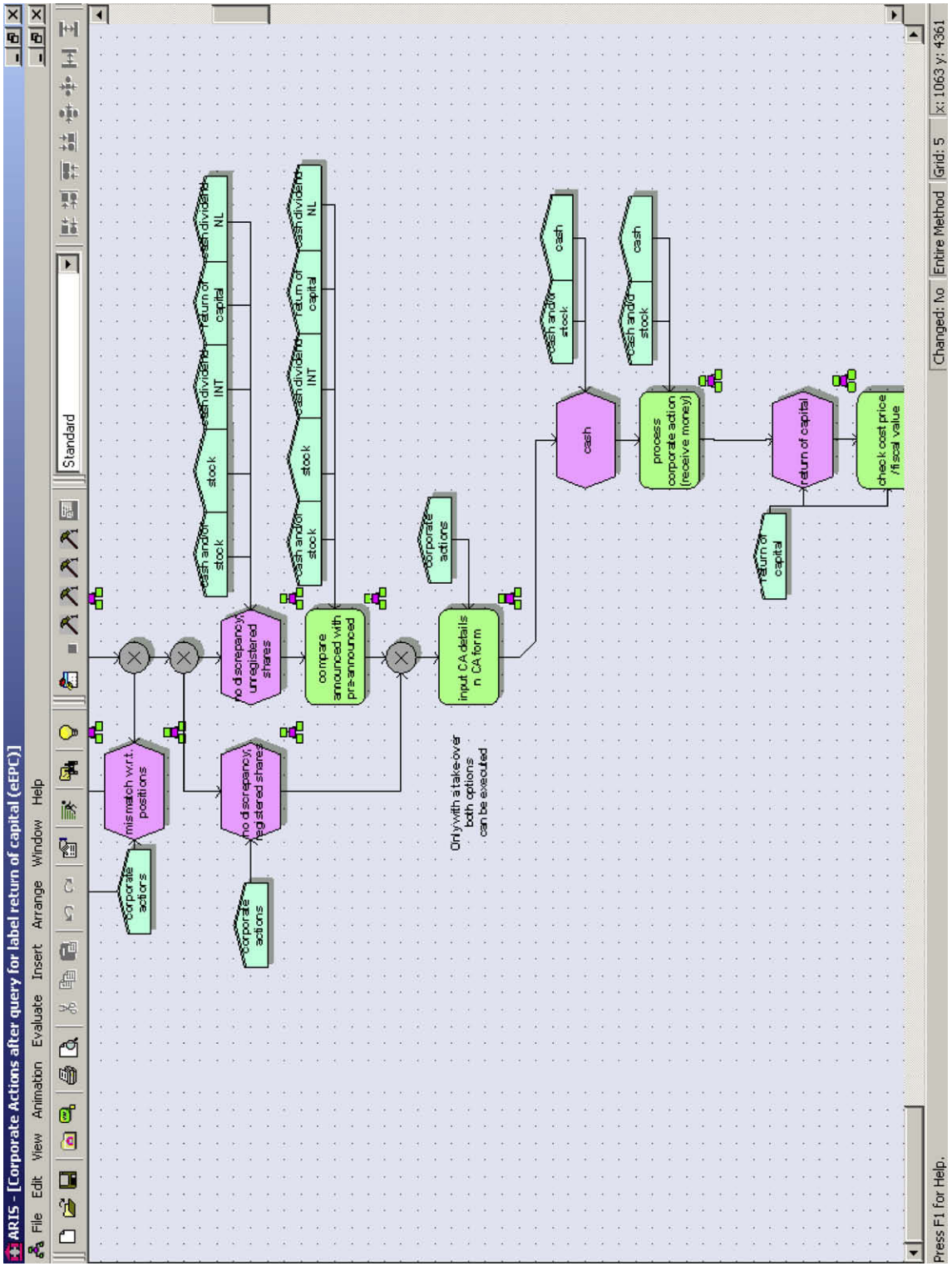

Fig. 12. Part of the extracted EPC.

tains the result of the extraction, is obtained. By performing extensive testing, we were able to demonstrate the correct functioning of the tool. Note that beyond its basic functionality, the developed tool can check whether a model conforms to the definition of aEPCs (e.g. whether no product labels are forgotten, no labels are used that are not part of the product hierarchy, etc.), and it can check whether an aEPC is correctly labeled with respect to a product hierarchy (e.g. whether a given product hierarchy matches a particular aEPC).

Returning to our example, the aEPC for the corporate actions process and its corresponding hierarchy was modeled in the ARIS Toolset. A screenshot of the aEPC can be seen in Fig. 11. Note that only part of the more than 80 functions and events of this process can be seen in this screenshot. In the middle of the screenshot, a function is present that is labeled 'Input CA details in CA form'. From the product label that is attached to it, it can be deduced that this function must be executed for any kind of corporate action. Immediately after this function, an inclusive OR connector (∨) can be distinguished. Its leftmost outgoing path leads to the 'stock' event, while its rightmost outgoing path leads to the 'cash' event. The product labels that are attached to these events show for which kinds of corporate actions these events are appropriate. What is interesting to note here is that the 'cash' label is only attached to the rightmost event.

As with all other aEPCs now in use within ING IM, extractions can be generated from this particular aEPC. Let us assume that we are interested in the process that is executed for the *return of capital* corporate action. With a *return of capital* some or all of the money that is invested in a company by an investor will be paid back to the investor by that company. So, a *return of capital* is a corporate action that delivers cash. This means that for the extracted model, we want to have *precisely* those functions and events that are relevant for *return of capital* itself, for cash specifically, and for corporate actions in general.

For the same part of the aEPC that is shown in Fig. 11, the resulting part of the extracted, singular process model for the *return of capital* product is shown in Fig. 12. In the middle of this screenshot, the function that is labeled 'Input CA details in CA form' can still be seen. This is correct, because it relates to a function that is relevant for all corporate actions, including *return of capital*. However, there is no logical connector that follows up this function as was the case in the aEPC of Fig. 11. Where there was an inclusive choice for a corporate action in this aEPC, there is no such choice for a specific *return of capital* in the model of Fig. 11. This is correct too: because the extracted model concerns cash but not stock, there is a direct connection from the 'Input CA details in CA form' function to the 'cash' event.

6.3. Evaluation

The provided, real-life example from ING IM shows that instead of creating fifteen separate models that describe the process for a single corporate action, it is feasible to create *one* aEPC that describes the process for *all* corporate actions. Clearly, this is much more efficient from a maintenance perspective.

To discuss the effectiveness of aEPCs on a more general level, we sat down together with three of the seven process analysts from ING IM's Information and Process Engineering department to discuss their experiences with the use of aEPCs in practice. All three modelers are familiar with the notation and have experience in applying it. We used a semi-structured interview scheme in this discussion, which covered the topics of (a) the *effectiveness* of the developed tool with respect to improved model management, (b) the *acceptance* of aEPCs among business users, and (c) the *ease* of creating and maintaining aEPCs.

All process analysts were of the opinion that aEPCs deliver a tangible contribution to decreasing the burden of managing a large set of process models. Its biggest value was seen in the aggregation of processes that span multiple departments, because in these situations process models are used most effectively to communicate with involved stakeholders. It was expected by one of the process analysts that the aEPCs would become even more useful if "downstream" and "upstream" process models within the same value chain would be further integrated in single models. This is one of the tasks that is currently under way.

With respect to aEPCs the process analysts were unanimous that such models would not at all be acceptable as communication means with business users. In other words, the facility to extract a specific model from an aEPC is a strict necessity. Furthermore, the process analysts indicated that the background of the business user in question probably plays a role in this too. The more someone is process-minded, in the sense that he or she commits to the idea that a process is a valuable entity to be coordinated, the more willing is such a person to study and discuss process models. A shortcoming of the current tool was also mentioned, i.e. that the deletion of irrelevant parts of an aEPC as a result of an extraction does not trigger an improved drawing of the remaining model. For the example in Fig. 12 it would improve the understanding and attractiveness of the model if the event 'cash' would be aligned under the preceding function, "Input CA details in CA form". This is a clear indication of another factor that presumably plays a role in the acceptance and use of process models.

Most surprisingly perhaps was the opinion of the process analysts about the ease of modeling an aEPC: this was not considered as a particularly difficult exercise. The process analyst that was responsible for integrating the 15 corporate actions into one model, for example, spent 1.5 days on this. To the specific question whether it is difficult to distinguish the relevant labels in a product hierarchy that matches an aEPC, the process analysts answered that these labels were often quite obvious. After all, they immediately relate to business concepts that are in use throughout ING IM to distinguish various tasks and duties. Finally, the process analysts acknowledged that it is time-consuming to keep aEPCs up-to-date, but not more so than this would be the case in general.

In conclusion, despite the fact that a considerable amount of labels is needed within an aEPC, our evaluation of its use in the business environment of ING IM suggests that they are valuable for improved model management, that the models extracted from it – by a push of a button – can be used well in the communication with business users, and that aEPCs are reasonably easy to create and maintain. Currently, ING IM is in the process of developing aggregate process models for all its processes.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

7. Related work

The management of process models as valuable business artifacts is a rather unexplored subject. The challenges with respect to creating, maintaining, and sharing process models have already been identified by Madhusudan et al. [25,26]. Where the emphasis in this work has been on the creation of new models, our work is complimentary in its focus on the use and disclosure of existing models.

The perspectives-based configurative reference process modeling approach by Becker et al. [5] is closely related to our idea of describing the process for several products with one process model and selecting its relevant parts. It focuses on adaptation mechanisms and proposes several mechanisms for automatically transforming a reference model into an individual model. However, to the best of our knowledge, no formalization has been given for these mechanisms. For our case, this hampers their translation towards tool support.

Additionally, the work of Rosemann and van der Aalst [35] pursues a reference model-driven approach with which individual models can be obtained, called configurable EPCs (C-EPCs). As we reflected on earlier in this paper, C-EPCs are rather difficult to understand, especially for end-users with a non-technical background. We could clearly establish this during the various discussions with business users at ING IM, where our case study has been carried out. It must be noted, however, that recent research efforts have been carried out to simplify the use of C-EPCs in configuring specific process models [22]. For our purposes, however, C-EPCs are less suitable as our focus is primarily on the communicative power of the process models itself. Note that also in [40] it is stressed that guidance is needed for obtaining individualized models from a reference model.

The wider research area on reference process models, which are intended to be configured in a specific setting such that they can lead to individualized process models, is too large to discuss in the context of this paper. In [35] a good overview of the various approaches can be found.

In this paper, we encountered the issue to resolve ambiguities from an aEPC in such a way that (a particular notion of) correctness is preserved. There have been previous approaches that also relate to transforming a process model while safe-guarding its initial correctness. In [32,34] the focus is on appropriate dynamic adaptions of process models, both at the type and instance level, to provide flexible support for workflow enactment. In [44] adaptions are discussed of process models that reside in large repositories to uphold their maintainability. Note that neither of these approaches puts an emphasis on the compositional or integrative nature of process models, as is the case in our work on aEPCs.

Composition and integration of process artifacts have been dealt with in other areas. A well-known issue in the field of workflow management is that execution is often distributed across various organizations and systems. In this context, different researchers have focused on so-called *private views* on the integrated workflow model and have proposed ways to manage the relationship between such views and the overall model, see e.g. [11,16,39]. Similarly, extensive attention has been devoted to the support of composing web services from more elementary services [7,8,45]. Typically, the work in the areas of distributed workflow execution and web services displays a high sophistication with respect to the automatic support for integrating and decomposing various models/views/web services. An important difference with our work is that our concern has been with models that are mainly used for *communication* purposes. This particular focus has motivated the design choices for our approach and the artifacts resulting from it. For example, in none of the work that is referenced a product hierarchy or a similar artifact plays any role, while it is essential in the use of aEPCs to provide meaningful model extractions for different business users.

8. Conclusion and outlook

In this paper, we described an approach to combine descriptions for various related process models into an aggregated process model. We formally introduced the aEPC and some supporting concepts to this end. Additionally, we presented an add-on for the ARIS Toolset which implements these concepts and illustrated how it is used in the context of model management issues in an industrial setting.

We argue that our approach is effective to simplify model management in practice, as it can contribute to a considerable reduction of the number of models that needs be maintained and updated. Also, the development of the aEPC is heavily guided by a desire to support human communication. This explains our focus on simple aids as labels to enrich the meaning of an aEPC (in comparison to simple EPCs), as well as our interest in the development of an algorithm to generate singular process models *on the fly*. At the same time, this paper illustrates how a simple design decision, i.e. to not allow for connector labels, may result in various correctness and ambiguity issues.

We believe that the aEPC and it relates concepts are widely applicable in settings where there is a need to reduce the amount of process models of a highly similar sort. ING IM is a typical example of such a setting, but many other organizations in the (financial) services domain can be imagined to have a comparable supply of models and a comparable need to efficiently and accurately manage their content.

Our approach is characterized by a number of limitations. First of all, the creation and evaluation of the various design alternatives for an aggregate process model has taken place in close collaboration with business professionals, but no rigorous empirical methods (e.g. surveys, experiments, etc.) have been employed to test the differences between these with respect to model comprehension. A more methodological evaluation of our approach could be the aim of future work. This

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

agenda fits nicely with recent work of ourselves and others in this field [10,27,33,42]. In particular, an experimental set-up as described in [37] seems suitable to approach this evaluation.

Second, our approach is specifically geared towards the use of EPCs. On the one hand, one can argue that this will cater for a large part of the business community, considering the extensive use of the ARIS Toolset. On the other hand, we have given little attention to discuss the introduced concepts for a wider set of modeling techniques. At first sight, it does not seem hard to generalize our approach to other activity-oriented modeling techniques – provided that such a technique incorporates a mechanism to use labels. For example, the `Protos` Toolset would be a good candidate [43].

Third, our approach does not take evolutionary development into account. Over time it can be needed to develop different variants of an aEPC. In our approach, if an aEPC is changed, the old variant is lost. In [41] this problem is discussed for reference models and a solution is presented for the version management of jointly designed reference models. A similar mechanism is highly relevant for our work as well.

A last limitation that is worth mentioning relates to our focus on *syntactical* correctness in this work, while many other properties that relate to process model quality can be distinguished. Some of the first additional relevant criteria that come to mind are *soundness* [2] and closely related properties [14,31]. In a general sense, the notion refers to the property of a process model to always terminate eventually in a desirable end situation. For C-EPCs in particular it has already been shown in [3] that when the reference process model is behaviorally sound, the individualized process models are guaranteed to be sound as well. Because of the formalization that underlies our work it will be relatively easy to extend our work in this direction.

Finally, we stress that our contribution addresses yet a small part of the issues that are involved with model management. For example, it is an open issue how the right balance can be determined between the complexity of an aggregated process model on the one hand and the number of models in a collection of process models on the other. It is safe to say that the management of process models as a discipline is at its infancy in comparison with fields like product and software management. Therefore, we hope that our work and experiences inspire others to pursue further research in this area.

References

[1] W.M.P. van der Aalst, Formalization and verification of event-driven process chains, Information and Software Technology 41 (10) (1999) 639–650.
[2] W.M.P. van der Aalst, Workflow verification: finding control-flow errors using Petri-net-based techniques, in: W.M.P. van der Aalst, J. Desel, A. Oberweis (Eds.), Business Process Management. Models, Techniques, and Empirical Studies, Lecture Notes in Computer Science, vol. 1806, Springer, 2000, pp. 161–183.
[3] W.M.P. van der Aalst, M. Dumas, F. Gottschalk, A.H.M. ter Hofstede, M. la Rosa, J. Mendling, Correctness-preserving configuration of business process models, in: J. Fiadeiro, P. Inverardi (Eds.), Fundamental Approaches to Software Engineering, Lecture Notes in Computer Science, vol. 4961, Springer, 2008, pp. 46–61.
[4] N.H. Bancroft, H. Seip, A. Sprengel, Implementing Sap R/3: How to Introduce a Large System into a Large Organization, second ed., Prentice Hall, Englewood Cliffs, 1997.
[5] J. Becker, P. Delfmann, R. Knackstedt, D. Kuropka, Konfigurative referenzmodellierung, in: J. Becker, R. Knackstedt (Eds.), Wissensmanagement mit Referenzmodellen, Konzepte für die Anwendungssystem-und Organisationsgestaltung, Physica-Verlag, Heidelberg, 2002, pp. 25–144.
[6] J. Becker, M. Rosemann, C. von Uthmann, Guidelines of business process modeling, in: W.M.P. van der Aalst, J. Desel, A. Oberweis (Eds.), Business Process Management. Models, Techniques, and Empirical Studies, Lecture Notes in Computer Science, vol. 1806, Springer, 2000, pp. 30–49.
[7] B. Benatallah, M. Dumas, Q.Z. Sheng, Facilitating the rapid development and scalable orchestration of composite web services, Distributed and Parallel Databases 17 (1) (2005) 5–37.
[8] B. Benatallah, Q.Z. Sheng, M. Dumas, The self-serve environment for web services composition, IEEE Internet Computing (2003) 40–48.
[9] G. Booch, J. Rumbaugh, I. Jacobson, et al, The Unified Modeling Language User Guide, Addison-Wesley, 1999.
[10] J. Cardoso, Process control-flow complexity metric: an empirical validation, in: Proceedings of IEEE International Conference on Services Computing (IEEE SCC 06), Chicago, USA, September 18–22, IEEE Computer Society, 2006, pp. 167–173.
[11] D.K.W. Chiu, K. Karlapalem, Q. Li, E. Kafeza, Workflow view based E-contracts in a cross-organizational E-services environment, Distributed and Parallel Databases 12 (2) (2002) 193–216.
[12] E.K. Clemons, M.E. Thatcher, C. Michael, Identifying sources of reengineering failures: a study of the behavioral factors contributing to reengineering risks, Journal of Management Information Systems 12 (2) (1995) 9–36.
[13] I. Davies, M. Rosemann, P. Green, Exploring proposed ontological issues of ARIS with different categories of modellers, in: 15th Australasian Conference on Information Systems, Hobart, 2004.
[14] J. Dehnert, P. Rittgen, Relaxed soundness of business processes, in: K.R. Dittrick, A. Geppert, M.C. Norrie (Eds.), Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001), Lecture Notes in Computer Science, vol. 2068, Springer, 2001, pp. 151–170.
[15] G.M. Giaglis, A taxonomy of business process modeling and information systems modeling techniques, International Journal of Flexible Manufacturing Systems 13 (2) (2001) 209–228.
[16] P. Grefen, H. Ludwig, S. Angelov, A three-level framework for process and data management of complex E-services, International Journal of Cooperative Information Systems 12 (4) (2003) 487–531.
[17] J.A. Gulla, T. Brasethvik, On the challenges of business modeling in large-scale reengineering projects, in: Peter P. Chen, David W. Embley, Jacques Kouloumdjian, Stephen W. Liddle, John F. Roddick (Eds.), Fourth International Conference on Requirements Engineering, IEEE, 2000, pp. 17–26.
[18] B.J. Hommes, V. van Reijswoud, Assessing the quality of business process modelling techniques, in: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 2000, pp. 1–10.
[19] G. Keller, M. Nüttgens, A.W. Scheer, Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)", Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992.
[20] M. Kesari, S. Chang, P.B. Seddon, A content-analytic study of the advantages and disadvantages of process modelling, in: J. Ang, S.-A. Knight (Eds.), Proceedings of the 14th Australasian Conference on Information Systems, Perth, 2003.
[21] P. Kueng, P. Kawalek, Process models: a help or a burden, in: J. Gupta (Ed.), Proceedings of the Americas Conference for Information Systems, 1997, pp. 676–678.
[22] M. La Rosa, J. Lux, S. Seidel, M. Dumas, A.H.M. ter Hofstede, Questionnaire-driven configuration of reference process models, in: Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE), Lecture Notes in Computer Science, vol. 4495, Springer, 2007, pp. 424–438.
[23] A. Lindsay, D. Downs, K. Lunn, Business processes – attempts to find a definition, Information and Software Technology 45 (15) (2003) 1015–1019.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

[24] W. Luo, Y. Tung, A framework for selecting business process modeling methods, Industrial Management and Data Systems 99 (7) (1999) 312–319.
[25] T. Madhusudan, A web services framework for distributed model management, Information Systems Frontiers 9 (1) (2007) 9–27.
[26] T. Madhusudan, J.L. Zhao, B. Marshall, A case-based reasoning framework for workflow model management, Data and Knowledge Engineering 50 (1) (2004) 87–115.
[27] J. Mendling, H.A. Reijers, J. Cardoso, What makes process models understandable?, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), Proceedings of the Fifth International Conference on Business Process Management (BPM 2007), Lecture Notes in Computer Science, vol. 4714, Springer, 2007, pp. 48–63.
[28] J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst, G. Neumann, Detection and prediction of errors in EPCs of the SAP reference model, Data and Knowledge Engineering 64 (1) (2008) 312–329.
[29] M.A. Ould, Business Processes: Modelling and Analysis for Re-engineering and Improvement, Wiley, 1995.
[30] A.N. Parr, G.G. Shanks, P. Darke, Identification of necessary factors for successful implementation of ERP systems, in: Proceedings of the IFIP TC8 WG8.2 International Working Conference on New Information Technologies in Organizational Processes, Kluwer, 1999, pp. 99–120.
[31] F. Puhlmann, M. Weske, Investigations on soundness regarding lazy activities, in: S. Dustdar, J.L. Fiadeiro, A. Sheth (Eds.), Proceedings of the Fourth International Conference on Business Process Management (BPM 2006), September, Lecture Notes in Computer Science, vol. 4102, Springer, 2006, pp. 145–160.
[32] M. Reichert, S. Rinderle, P. Dadam, On the common support of workflow type and instance changes under correctness constraints, in: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, Lecture Notes in Computer Science, vol. 2888, Springer, 2003, pp. 407–425.
[33] H.A. Reijers, I.T.P. Vanderfeesten, Cohesion and coupling metrics for workflow process design, in: J. Desel, B. Pernici, M. Weske (Eds.), Proceedings of the Second International Conference on Business Process Management (BPM 2004), Lecture Notes in Computer Science, vol. 3080, Springer, 2004, pp. 290–305.
[34] S. Rinderle, M. Reichert, P. Dadam, Evaluation of correctness criteria for dynamic workflow changes, in: W.M.P. van der Aalst, A.H.M. ter Hofstede, M. Weske (Eds.), Proceedings of the First International Conference on Business Process Management (BPM 2003), Lecture Notes in Computer Science, vol. 2678, Springer, 2003, pp. 41–57.
[35] M. Rosemann, W.M.P. van der Aalst, A configurable reference modelling language, Information Systems 32 (1) (2007) 1–23.
[36] W. Sadiq, S. Sadiq, K. Schulz, Model driven distribution of collaborative business processes, in: IEEE International Conference on Services Computing, SCC, 2006, pp. 281–284.
[37] K. Sarshar, P. Loos, Comparing the control-flow of EPC and Petri net from the end-user perspective, in: W.M.P. van der Aalst, B. Benatallah, F. Casati, F. Curbera (Eds.), Proceedings of the Third International Conference on Business Process Management (BPM 2005), Lecture Notes in Computer Science, vol. 3649, Springer, 2005, pp. 434–439.
[38] A.W. Scheer, ARIS Business Process Modelling, Springer, 2000.
[39] K.A. Schulz, M.E. Orlowska, Facilitating cross-organisational workflows with a workflow view approach, Data and Knowledge Engineering 51 (1) (2004) 109–147.
[40] P. Soffer, I. Reinhartz-berger, A. Sturm, Facilitating reuse by specialization of reference models for business process design, in: B. Pernici, J.A. Gulla (Eds.), Proceedings of Workshops and Doctoral Consortium of the 19th International Conference on Advanced Information Systems Engineering (BPMDS Workshop), vol. 1, 2007.
[41] O. Thomas, Joint reference modeling: collaboration support through version management, in: R.H. Sprague (Ed.), Proceedings of the 40th Annual Hawaii International Conference on System Sciences, January, IEEE Computer Society Press, Big Island, Hawaii, 2007.
[42] I. Vanderfeesten, J. Cardoso, J. Mendling, H.A. Reijers, W.M.P. van der Aalst, Quality metrics for business process models, in: 2007 BPM and Workflow Handbook, Future Strategies Inc., 2007, pp. 179–190.
[43] H.M.W. Verbeek, M. van Hattem, H.A. Reijers, W. de Munk, Protos 7.0: simulation made accessible, in: Applications and Theory of Petri Nets, Lecture Notes in Computer Science, vol. 3536, Springer, 2005, pp. 465–474.
[44] B. Weber, M. Reichert, Refactoring process models in large process repositories, in: Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE), Lecture Notes in Computer Science, vol. 5074, Springer, 2008, pp. 124–139.
[45] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, H. Chang, QoS-aware middleware for web services composition, IEEE Transactions on Software Engineering (2004) 311–327.

H.A. Reijers is an Assistant Professor of Information Systems in the School of Industrial Engineering at Eindhoven University of Technology and affiliated as staff member to the TiasNimbas Business School in Tilburg. He earned a Ph.D. in Computer Science from Eindhoven University of Technology in 2002, while he worked as a manager in the consultancy practice of Deloitte. His research interests are in business process modeling, workflow management technology, and discrete event simulation. He is founder and member of the Dutch BPM-Forum, a platform for knowledge exchange between industry and academia related to business process optimization. He has published in a variety of academic journals including Information Systems, Journal of Management Information Systems, International Journal of Cooperative Systems, Computers and Industry, Computer Supported Cooperative Work, and Omega.

R.S. Mans received his M.Sc. degree in Computer Science from Eindhoven University of Technology in 2006. At present he is a Ph.D. candidate at the Department of Technology Management at the same university. The topic of the Ph.D. project is on the use of workflow technology in the healthcare domain. His research interests include workflow management, process mining, and discrete event simulation.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004

R.A. van der Toorn is manager of the Information and Process Engineering department of ING Investment Management Europe. In that role he is responsible for both Enterprise Architecture and Business Process Management. Before he joined ING Investment Management in 2004, he worked for over 10 year as a management consultant. First for the software company Sogeti and later for Deloitte & Touche. For Deloitte he worked for various clients on various assignments considering business and information analysis, process redesign and enterprise architectures. Simultaneously he worked for the Technische Universiteit Eindhoven (TU/e) as a Ph.D. from 1997 to 2004. His research at TU/e focused on the scientific foundations of the things he applied in practice at Deloitte.

Please cite this article in press as: H.A. Reijers et al., Improved model management with aggregated business process models, Data Knowl. Eng. (2008), doi:10.1016/j.datak.2008.09.004