

Beyond Control-Flow: Extending Business Process Configuration to Resources and Objects

M. La Rosa¹, M. Dumas^{1,2}, A.H.M. ter Hofstede¹, J. Mendling¹, and F. Gottschalk³

¹ Queensland University of Technology, Australia
{m.larosa, j.mendling, m.dumas, a.terhofstede}@qut.edu.au

² University of Tartu, Estonia
marlon.dumas@ut.ee

³ Eindhoven University of Technology, The Netherlands
f.gottschalk@tm.tue.nl

Abstract. A configurable process model is an integrated representation of multiple variants of a business process. It is designed to be individualized to meet a particular set of requirements. As such, configurable process models promote systematic reuse of proven or common practices. Existing notations for configurable process modeling focus on capturing tasks and control-flow dependencies, neglecting equally important aspects of business processes such as data flow, material flow and resource management. This paper fills this gap by proposing an integrated meta-model for configurable processes with advanced features for capturing resources involved in the performance of tasks (through task-role associations) as well as flow of data and physical artifacts (through task-object associations). Although embodied as an extension of a popular process modeling notation, namely EPC, the meta-model is defined in an abstract and formal manner to make it applicable to other notations.

Key words: Process model, configuration, resource, object flow

1 Introduction

Reference process models such as the Supply Chain Operations Reference (SCOR) model [22] or the SAP Reference Model [6], capture recurrent business operations in their respective domains. They are packaged as libraries of models in several business process modeling tools and are used by analysts to derive process models for specific organizations or IT projects (a practice known as *individualization*) as an alternative to designing process models from scratch.

Reference process models in commercial use lack a representation of variation points and configuration decisions. As a result, analysts are given little guidance as to which model elements need to be removed, added or modified to address a given requirement. This shortcoming is addressed by the concept of *configurable process model* [19], which captures process variants in an integrated manner. This concept is a step forward towards systematic reuse of (reference) process models. However, existing configurable process modeling languages focus on the control-flow perspective and fail to capture resources, data and physical objects involved in the performance of tasks.

This paper extends a configurable process modeling notation, namely Configurable Event-driven Process Chains (C-EPCs), with notions of roles and objects. The proposed extension supports the representation of a range of variations in the way roles and objects are associated with tasks. The proposal is formally defined and the formalization is used to formulate conditions that ensure the syntactic correctness of the configured models, which are obtained via a derivation algorithm. The paper also explores interplays that occur across the control-flow, object flow and resource modeling perspectives during individualization. The proposal has been applied to a comprehensive case study in the film industry, which is used as an example throughout the paper.

The rest of the paper is structured as follows. Section 2 reviews previous work related to configurable process modeling and modeling of object flow and resources in business processes. Section 3 introduces the working example and uses it to illustrate a meta-model that extends EPCs with resource and object flow modeling. This meta-model is formalized in Section 4. Next, Section 5 explores the configuration of process models along the resource and object flow perspectives, leading to a formal meta-model of a fully-featured C-EPC notation in Section 6.

2 Background and Related Work

Business process models can be seen from a number of perspectives, including the control-flow, the data and the resource perspectives [11]. In this paper, we are concerned with defining a process modeling notation that covers all three perspectives while incorporating features for configurable modeling.

A common approach to capture resources in process models is to associate a role, a capability and/or an organizational group to each task [1]. In several flowchart-like notations, such as UML activity diagrams [7] or BPMN [23], this association is encoded by means of *swimlanes*. Each task (or activity) is associated to a swimlane which may represent a role or an organizational unit. UML Activity Diagrams (ADs) allow multiple swimlanes (or *partitions*) to be associated to an activity. An activity can only be performed by a resource that belongs to all the partitions associated to it. In (extended) EPCs [21], symbols denoting roles or organizational units can be freely attached to tasks, in any combination but with no specific semantics. Extended EPCs (eEPCs) have been partially formalized for simulation purposes in [10]. In this paper, we define more sophisticated role-based resource modeling features than in eEPCs and we layer configuration features on top of them. In this paper, we define more sophisticated role-based resource modeling features and we layer configuration features on top of them. The features we define go beyond those found in UML ADs and BPMN.

A detailed approach to capturing resources in process models, with an emphasis on resource allocation is outlined by zur Mühlen [15], while Russell et al. [20] identify a set of resource patterns describing various types of associations between resources and tasks. However, these patterns are not embodied in a process modeling notation. Data flow and resource modeling for business processes has also been studied from the perspective of access control. Ferraiolo et al. [8] outline a reference model of well-accepted mechanisms for role-based access control, while Bertino et al. formalize authorization

constraints and discuss their specification and enforcement in workflow management systems [5]. This body of work is complementary to our proposal.

Meanwhile, the flow of data and physical artifacts (e.g. paper documents or production materials) is generally captured by associating objects to tasks. UML ADs support the association of object nodes to tasks to denote inputs and outputs. UML ADs also provide features for modeling streams – object nodes that can store multiple objects at once. A subset of these features are also found in BPMN. In UML ADs, one can associate multiple objects as input or as output of an activity. The execution of an activity consumes one object from each of the activity’s input object nodes and produces one object in each of its output object nodes. In BPMN, the semantics of such constructions is underspecified. The same applies to (extended) EPCs, which allow multiple data nodes to be connected to a function but with no specific semantics. In this paper, we propose a more fine-grained approach to object flow modeling as well as mechanisms to define variability in the way object nodes are associated with tasks.

Languages for executable process modeling, such as ADEPT_{flex} [17], BPEL [3], or YAWL [2], generally rely on global variables to capture data flow. These languages are also concerned with the definition of data mappings between global variables and task input/output parameters. Such mappings are also used in formal system specifications based on (Colored) Petri nets [12]. In this paper, our aim is to define configurable process models for analysis and design. As such, our proposal does not cover aspects such as data mappings which are relevant at the implementation level.

Research on configurable business process models has focused on mechanisms for capturing variability along the control-flow perspective. Rosemann & van der Aalst [19] put forward the C-EPC notation where tasks can be switched on or off and routing connectors can be made configurable and linked through configuration requirements. Becker et al. [4] introduce an approach to hide element types in EPCs for configuration purposes. Although the emphasis is on tasks and control-flow connectors, this approach can also be used to show or hide resource or data types. However, this only affects the view on the EPC, not its underlying behavior. Also, this approach does not enable fine-grained configuration of task-role and task-object associations (beyond hiding).

In previous work, we have investigated a basic set of process configuration operators based on skipping and blocking of tasks [9]. In [13] we defined an algorithm to derive syntactically correct EPCs from configured C-EPCs. In [16] we identify a generic process for the task of model-driven Enterprise Systems configuration and describe it in a notation-independent manner. We have also investigated the use questionnaires to help domain experts to configure C-EPCs [18].

In this paper, we use EPCs as a base notation to define variability mechanisms along the data and resource perspectives. Three reasons underpin this choice. First, EPCs are widely used for reference process modeling (cf. the SAP reference model). Secondly, although lacking a precise definition, EPCs provide basic features for associating data nodes and roles to tasks. Finally, this choice allows us to build on top of the existing formal definition of the C-EPC notation. Nonetheless, we define our extensions in an abstract manner so that they are applicable beyond the scope of EPCs.

3 Working Example

The working example in Fig. 1 is an extract of a broader reference process model on music and sound editing for screen post-production. The model has been developed and validated in collaboration with subject-matter experts of the Australian Film Television & Radio School.¹ The reason for choosing this process is the high level of creativity, and thus of variability, that characterizes this domain. Indeed, the whole editing phase can be radically different if the screen project aims to produce a documentary (usually without music) or a silent movie (without spoken dialogs). Below we describe the process as if it were non-configurable, to illustrate how we capture resources and objects participating in an EPC process. The configuration aspects will be addressed later on, so for now we ignore the meaning of the thick border of some elements in the picture.

EPC's main components are events, functions, control-flow connectors, and arcs linking these elements. Events model triggers or conditions, functions correspond to tasks and connectors denote splits and joins of type AND, OR or XOR. We extend these concepts by associating roles and objects to functions. Roles, depicted on a function's left hand, capture organizational resources able to perform functions. For simplicity, in the example roles only refer to human resources. Input (output) objects, depicted on a function's right hand, capture physical or software artifacts required/produced by the function. The first function is Spotting session, which starts once the shooting has completed. Roles and objects are linked to functions either directly or via a connector. For example, the OR-join between Composer and Sound Designer indicates that at least one of these two roles is required to perform this activity (roles are linked to functions via directed arcs). Composer is needed if the project features music, Sound Designer is needed if the project features sound, which consists of dialogs, effects (FX) and/or atmospheres (atmos). Composer and Sound Designer hold a Spotting session to decide on the cues for music and sound, based on the screening of the Picture cut. Picture cut is thus an input object (linked to the function via a directed arc), while the cues are output objects (linked via an arc directed from the function to the object).

An OR-split connecting the cues indicates that at least one set of cues is produced as a result of a Spotting session, depending on the type of project. The session may be supervised by at least two roles among Producer, Director and Assistant Director, that have creative authority in the project. These roles are linked together by a *range* connector. This connector indicates the upper and lower bound for number of elements (i.e. roles) that are required (where k refers to the indegree for a join or to the outdegree for a split; in this case $k = 3$). A range connector can be used for roles and objects.

Once the cues are available, the design of music and/or sound starts. In the former, the Composer records the project's Music tracks (an output) following the Music cues and using the Picture cut as a reference (an AND-join connects these two inputs). A Temp music file may also be produced at this stage. This object is linked to the function via a dashed arc, which indicates that an object, a role, or a combination thereof is optional, whereas a full arc indicates mandatoriness. Sound design is usually more complex as it involves the recording of the Dialog, FX and/or Atmos tracks, according to the respective cues on the Picture cut. The Editor or the Sound Designer (at least

¹ The school's web-site can be accessed at www.aftrs.edu.au

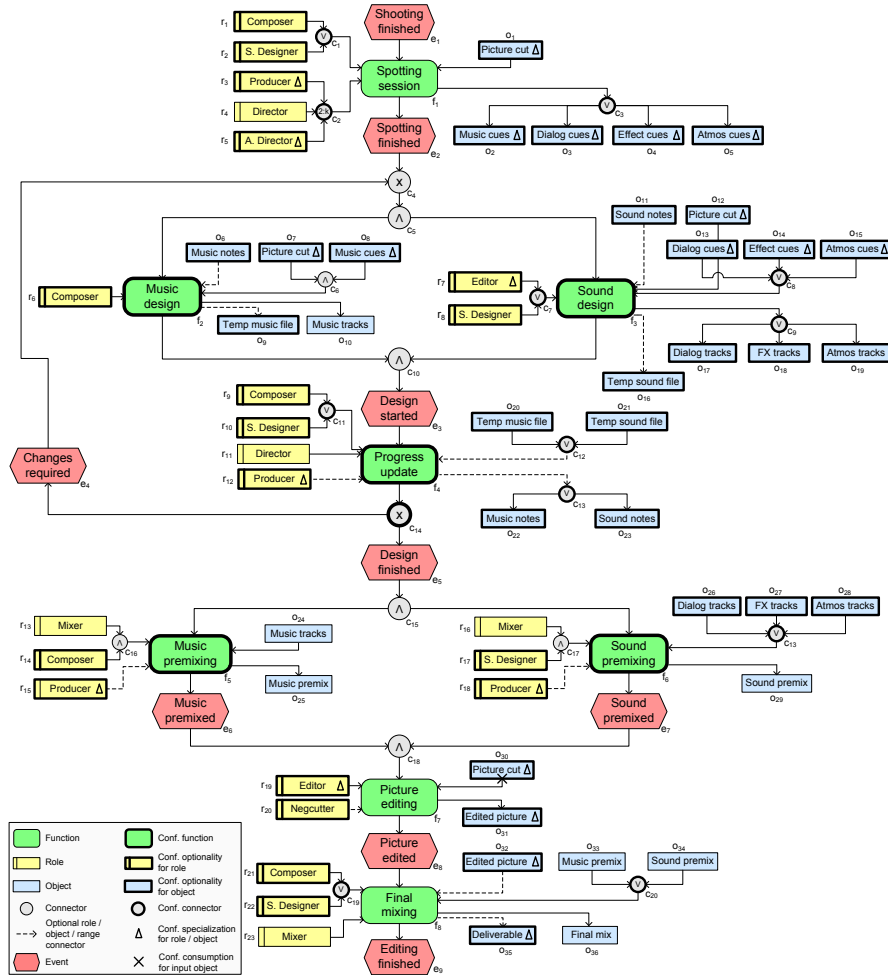


Fig. 1. The reference process model for audio editing

one) are responsible for this task. Similarly to Music design, a Temp sound file may also be produced. Afterwards, the Composer and/or the Sound Designer provide the Director and usually the Producer with an update on the work-in-progress. Producer is an optional role. At least a mandatory role is to be assigned to each function (either directly or via a mandatory connector) to ensure its execution. Temp files may be used by the Composer and by the Sound Designer as a guide for the Progress update (the OR-join between these two objects is thus optional). Generally, the result of this task is a set of notes describing the changes required; sometimes, however, the Composer or the Sound Designer may prefer not to take any notes. If changes are needed, the Music and Sound design can be repeated as specified by the loop in the model. In this case, the notes can be used as input to these tasks. Upon completion of the design phase, the Composer and the Mixer mix the Music tracks into a Music premix if the project

has music, while the Sound Designer and the Mixer mix the Sound tracks into a Sound premix if the project has sound. The Producer may supervise both mixings. In Picture editing, the Picture cut is edited by an Editor, while a Negcutter is required if the cut is on Film. The cross below ‘Picture cut’ indicates that the object is consumed by the function and is no longer available afterwards. The process finishes with the Final mixing, where the Sound Designer and/or the Composer with the Mixer release a Final mix using the available Premixes. They may also release a Deliverable by overlaying the Final mix onto the Edited picture, if a demo of the picture with the integrated audio is required at this stage.

Beside the process model, we use a ‘hierarchy model’ to represent all the roles and objects referred to by the nodes of the process model. For example, in the editing process there are five nodes for the role Producer and four for the object picture cut. A hierarchy model also captures the specializations that can be associated to a role (object), by means of a specialization relation. Fig. 2 shows the hierarchy models for the roles and objects of the editing process, where the specialization relation is depicted by an empty arrow linking a special role (object) to its generalization. Typically, for a role this relation represents a separation of tasks among its specializations (e.g., Executive Producer, Line Producer and Co-Producer share the Producer’s duties); where a role can be covered by one or more physical persons. For an object, it represents a set of subtypes (e.g. 16mm and 35mm are two types of Film). The meaning of this relation depends on the process’s domain. The hierarchy models will be used later on in the configuration of the process model.

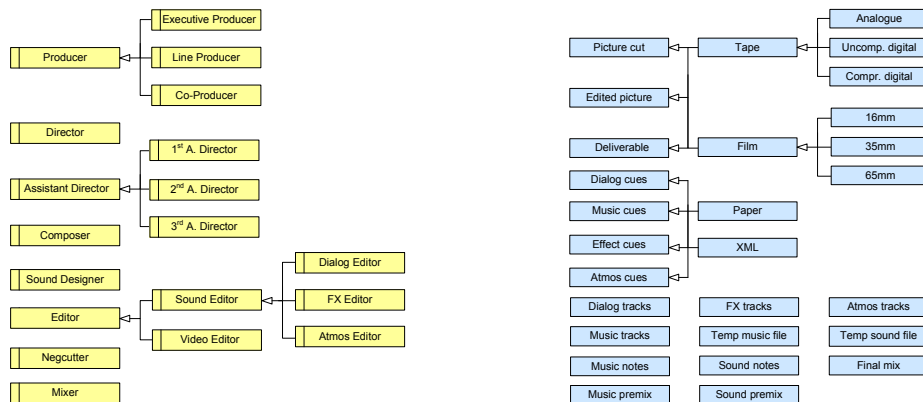


Fig. 2. The role-hierarchy model and the object-hierarchy model for the process of Fig. 1

4 Integrated Business Process Model

This section formalizes the notions discussed so far. The formalization allows us to convey the ideas in an unambiguous way and paves the way to the definition of the

configurable process model. We first define the concept of role- and object-hierarchy model as a set of roles (objects) for which a specialization relation is defined.

Definition 1 (Role-hierarchy Model). A role-hierarchy model is a tuple $Rh = (R, \overset{R}{\leftarrow})$, where:

- R is a finite, non-empty set of roles,
- $\overset{R}{\leftarrow} \subseteq R \times R$ is the specialization relation on R ($\overset{R}{\leftarrow}$ is transitive, reflexive, acyclic²).

Definition 2 (Object-hierarchy Model). An object-hierarchy model is a tuple $Oh = (O, \overset{O}{\leftarrow})$, where:

- O is a finite, non-empty set of objects, i.e. physical or software artifacts,
- $\overset{O}{\leftarrow} \subseteq O \times O$ is the specialization relation on O ($\overset{O}{\leftarrow}$ is transitive, reflexive, acyclic).

If $x_1 \overset{R/O}{\leftarrow} x_2$, we say x_1 is a *generalization* of x_2 and x_2 is a *specialization* of x_1 ($x_1 \neq x_2$). For example, Dialog Editor is a specialization of Editor.

The sets of roles and objects from the hierarchy-models are used together with a set of functions, in the definition of integrated EPC (iEPC). This definition extends the definition of EPC from [19], which focuses on the control-flow only, by providing a precise representation of resources (roles) and objects participating in the process.

In an iEPC each node represents an instance of a function, role or object. The range connector is modeled by a pair of natural numbers: lower bound (n) and upper bound (m). Indeed, an AND, OR and XOR correspond to a range connector resp. with $n = m = k$, with $n = 1, m = k$ and with $n = m = 1$. So we do not need to model the logic operators with separate connectors for roles and objects, although they can be graphically represented with the traditional EPC notation, as in Fig. 1. For the sake of keeping the model consistent with previous formalizations of EPC, the range connector is not allowed in the control-flow. Minimal effort would however be required to add this construct. The optionality of roles, objects and range connectors, shown in the process as a property of the arc that links the node with the function, is modeled in iEPC as an attribute of the nodes. The consumption of input objects is modeled in the same way.

Definition 3 (iEPC). Let F be a set of functions, $Rh = (R, \overset{R}{\leftarrow})$ be a role-hierarchy model and $Oh = (O, \overset{O}{\leftarrow})$ be an object-hierarchy model. An integrated EPC over F, Rh, Oh is a tuple $iEPC_{F,Rh,Oh} = (E, F_N, R_N, O_N, nm, C, A, L)$, where:

- E is a finite, non-empty set of events;
- F_N is a finite, non-empty set of function nodes for the process;
- R_N is a finite, non-empty set of role nodes for the process;
- O_N is a finite set of object nodes for the process;
- $nm = nf \cup nr \cup no$, where:
 - $nf \in F_N \rightarrow F$ assigns each function node to a function;
 - $nr \in R_N \rightarrow R$ assigns each role node to a role;
 - $no \in O_N \rightarrow O$ assigns each object node to an object;
- $C = C_{CF} \cup C_R \cup C_{IN} \cup C_{OUT}$ is a finite set of logical connectors, where:
 - C_{CF} is the set of control-flow connectors,

² no cycles of length greater than one

- C_R is the set of range connectors for role nodes (role connectors),
- C_{IN} is the set of range connectors for input nodes (input connectors),
- C_{OUT} is the set of range connectors for output nodes (output connectors),
- where C_{CF}, C_R, C_{IN} and C_{OUT} are mutually disjoint;
- $A = A_{CF} \cup A_R \cup A_{IN} \cup A_{OUT}$ is a set of arcs, where:
 - $A_{CF} \subseteq (E \times F_N) \cup (F_N \times E) \cup (E \times C_{CF}) \cup (C_{CF} \times E) \cup (F_N \times C_{CF}) \cup (C_{CF} \times F_N) \cup (C_{CF} \times C_{CF})$ is the set of control-flow arcs,
 - $A_R \subseteq (R_N \times F_N) \cup (R_N \times C_R) \cup (C_R \times F_N)$ is the set of role arcs,
 - $A_{IN} \subseteq (O_N \times F_N) \cup (O_N \times C_{IN}) \cup (C_{IN} \times F_N)$ is the set of input arcs,
 - $A_{OUT} \subseteq (F_N \times O_N) \cup (F_N \times C_{OUT}) \cup (C_{OUT} \times O_N)$ is the set of output arcs,
 - where A_R, A_{IN} and A_{OUT} are intransitive relations;
- $L = l_C^T \cup l_C^N \cup l_C^M \cup l_R^M \cup l_O^M \cup l_O^U$ is a set of label assignments, where:
 - $l_C^T \in C_{CF} \rightarrow \{AND, OR, XOR\}$ specifies the type of control-flow connector,
 - $l_C^N \in (C_R \cup C_{IN} \cup C_{OUT}) \rightarrow \mathbb{N} \times (\mathbb{N} \cup \{k\}) \cup \{(k, k)\}$, specifies lower bound and upper bound of the range connector,
 - $l_C^M \in (C_R \cup C_{IN} \cup C_{OUT}) \rightarrow \{MND, OPT\}$ specifies if a role connector, an input connector or an output connector is mandatory or optional,
 - $l_R^M \in R_N \rightarrow \{MND, OPT\}$ specifies if a role node is mandatory or optional;
 - $l_O^M \in O_N \rightarrow \{MND, OPT\}$ specifies if an object node is mandatory or optional;
 - $l_O^U \in O_N^{IN} \rightarrow \{USE, CNS\}$ specifies if an input object node is used or consumed, where $O_N^{IN} = \text{dom}(A_{IN}) \cap O_N$.

Let $l_C^N(c) = (n, m)$ for all $c \in C \setminus C_{CF}$. Then $lwb(c) = n$ and $upb(c) = m$ refer to lower bound and upper bound of c . If F, Rh and Oh are clear from the context, we drop the subscript from $iEPC$. Also, we call all the function nodes, role nodes and object nodes simply as functions, roles and objects, wherever this does not lead to confusion.

The following notation is introduced to allow a more concise characterization of $iEPCs$.

Definition 4 (Auxiliary sets, functions and predicates). For an $iEPC$ we define the following subsets of its nodes, functions and predicates:

- $N_{CF} = E \cup F_N \cup C_{CF}$, as its set of control-flow nodes;
- $N_R = F_N \cup R_N \cup C_R$, as its set of role nodes;
- $N_{IN} = F_N \cup O_N^{IN} \cup C_{IN}$, as its set of input nodes;
- $N_{OUT} = F_N \cup O_N^{OUT} \cup C_{OUT}$, as its set of output nodes, where $O_N^{OUT} = \text{dom}(A_{OUT}) \cap O_N$;
- $N = N_{CF} \cup N_R \cup N_{IN} \cup N_{OUT}$, as its set of nodes;
- $\forall n \in N_\alpha \overset{\alpha}{\bullet} n = \{x \in N_\alpha \mid (x, n) \in A_\alpha\}$, as the α -preset of n , $\alpha \in \{CF, R, IN, OUT\}$;
- $\forall n \in N_\alpha n \overset{\alpha}{\bullet} = \{x \in N_\alpha \mid (n, x) \in A_\alpha\}$, as the α -postset of n , $\alpha \in \{CF, R, IN, OUT\}$;
- $\bullet X = \bigcup_{x \in X, \alpha \in \{(CF), R, (IN), (OUT)\}} \overset{\alpha}{\bullet} x$, as the union on the presets of X ;
- $X \bullet = \bigcup_{x \in X, \alpha \in \{(CF), R, (IN), (OUT)\}} x \overset{\alpha}{\bullet}$, as the union on the postsets of X ;
- $E_s = \{e \in E \mid | \overset{CF}{\bullet} e | = 0 \wedge | e \overset{CF}{\bullet} | = 1\}$ as the set of start events;
- $E_e = \{e \in E \mid | \overset{CF}{\bullet} e | = 1 \wedge | e \overset{CF}{\bullet} | = 0\}$ as the set of end events;
- $C_{CF}^S = \{c \in C_{CF} \mid | \overset{CF}{\bullet} c | = 1 \wedge | c \overset{CF}{\bullet} | > 1\}$ as the set of control-flow split connectors;
- $C_{CF}^J = \{c \in C_{CF} \mid | \overset{CF}{\bullet} c | > 1 \wedge | c \overset{CF}{\bullet} | = 1\}$ as the set of control-flow join connectors;
- $link^\alpha(x, y) = \begin{cases} (y, x) \in A_R, & \text{if } \alpha = R, \text{ returns the role arc from } y \text{ to } x, \\ (y, x) \in A_{IN}, & \text{if } \alpha = IN, \text{ returns the input arc from } y \text{ to } x, \\ (x, y) \in A_{OUT}, & \text{if } \alpha = OUT, \text{ returns the output arc from } x \text{ to } y; \end{cases}$

- $degree(x) = \begin{cases} | \overset{R}{\bullet} x |, & \text{if } x \in C_R, \text{ returns the indegree of a role connector;} \\ | \overset{IN}{\bullet} x |, & \text{if } x \in C_{IN}, \text{ returns the indegree of an input connector;} \\ | x \overset{OUT}{\bullet} |, & \text{if } x \in C_{OUT}, \text{ returns the outdegree of an output connector;} \end{cases}$
- $p = \langle n_1, n_2, \dots, n_k \rangle$ is a control-flow path such that $(n_i, n_{i+1}) \in A_{CF}$ for $1 \leq i \leq k-1$. For short, we indicate that p is a path from n_1 to n_k as $p : n_1 \hookrightarrow n_k$. Also, $P(p) = \{n_1, \dots, n_k\}$ indicates the alphabet of p .

It follows that $\forall f \in F_N \ |f \overset{R}{\bullet}| = 0 \wedge |f \overset{IN}{\bullet}| = 0 \wedge | \overset{OUT}{\bullet} f | = 0$; $\forall r \in R_N \ | \overset{R}{\bullet} r | = 0$; $\forall o \in O_N \ | \overset{IN}{\bullet} o | = 0 \wedge | o \overset{OUT}{\bullet} | = 0$.

In the remainder, we assume an iEPC to satisfy the following syntactical requirements.

Definition 5 (Syntactically Correct iEPC). An iEPC is syntactically correct if it fulfills the following requirements:

1. iEPC is a directed graph such that every control-flow node is on a control-flow path from a start to an end event:
let $e_s \in E_s$ and $e_e \in E_e$, then $\forall n \in N_{CF} \ \exists_{p \in N_{CF}^+, p: e_s \hookrightarrow e_e} [n \in P(p)]$.
2. There is at least one start event and one end event in iEPC: $|E_s| > 0$ and $|E_e| > 0$.
3. Events have at most one incoming and one outgoing control-flow arc:
 $\forall e \in E \ [| \overset{CF}{\bullet} e | \leq 1 \wedge | e \overset{CF}{\bullet} | \leq 1]$.
4. Functions have exactly one incoming and one outgoing control-flow arc:
 $\forall f \in F_N \ [| \overset{CF}{\bullet} f | = | f \overset{CF}{\bullet} | = 1]$.
5. Control-flow connectors have one incoming and multiple outgoing arcs or vice versa:
 $\forall c \in C_{CF} \ [| \overset{CF}{\bullet} c | = 1 \wedge | c \overset{CF}{\bullet} | > 1 \vee (| \overset{CF}{\bullet} c | > 1 \wedge | c \overset{CF}{\bullet} | = 1)]$, (split, join),
Role connectors have multiple incoming arcs and exactly one outgoing arc:
 $\forall c \in C_R \ [| \overset{R}{\bullet} c | > 1 \wedge | c \overset{R}{\bullet} | = 1]$, (join),
Input connectors have multiple incoming arcs and exactly one outgoing arc:
 $\forall c \in C_{IN} \ [| \overset{IN}{\bullet} c | > 1 \wedge | c \overset{IN}{\bullet} | = 1]$, (join),
Output connectors have exactly one incoming arc and multiple outgoing arcs:
 $\forall c \in C_{OUT} \ [| \overset{OUT}{\bullet} c | = 1 \wedge | c \overset{OUT}{\bullet} | > 1]$, (split).
6. Roles have exactly one outgoing arc:
 $\forall r \in R_N \ |r \overset{R}{\bullet}| = 1$.
7. Objects have exactly one outgoing input arc or one incoming output arc:
 $\forall o \in O_N \ [(|o \overset{IN}{\bullet}| = 1 \wedge | \overset{OUT}{\bullet} o | = 0) \vee (|o \overset{IN}{\bullet}| = 0 \wedge | \overset{OUT}{\bullet} o | = 1)]$.
8. Functions are linked to at least a mandatory role or a mandatory role connector:
 $\forall f \in F_N \ [\exists_{r \in \overset{R}{\bullet} f} [l_R^M(r) = MND] \vee \exists_{c \in \overset{R}{\bullet} f} [l_C^M(c) = MND]]$, it follows that $| \overset{R}{\bullet} f | > 0$.
9. Roles and objects linked to connectors are mandatory:³
 $\forall r \in R_N \ [r \in \text{dom}((R_N \times C_R) \cap A_R) \Rightarrow l_R^M(r) = MND]$,
 $\forall o \in O_N \ [o \in \text{dom}((O_N \times C_{IN}) \cap A_{IN}) \Rightarrow l_O^M(o) = MND]$,
 $\forall o \in O_N \ [o \in \text{dom}((C_{OUT} \times O_N) \cap A_{OUT}) \Rightarrow l_O^M(o) = MND]$.
10. Upper bound and lower bound of range connectors are restricted as follows:
 $\forall c \in C_R \cup C_{IN} \cup C_{OUT} \ [1 \leq \text{lwb}(c) \leq \text{upb}(c) \wedge (\text{lwb}(c) \leq \text{degree}(c) \vee \text{upb}(c) = k)]$,
where $n \leq m$ iff $(n \leq m) \vee (m = k) \vee (n = m = k)$.

³ The optionality of a group of roles/objects linked by a range connector is modeled by making the connector optional

Clearly, the editing process model of Fig. 1 fulfills the above requirements. However, Def. 5 does not prevent behavioral issues (e.g. deadlocks) that may occur at run-time. It is outside the scope of the paper to provide a formal definition of the dynamic behavior of iEPCs, as we only consider structural correctness in the context of configuration. For completeness, we briefly discuss its semantics.

The dynamic behavior of iEPC has to take into account not only the routing rules of the control-flow, but also the availability of the resources and the existence of the objects present in the model. A state of the execution of an iEPC can be identified by a marking of tokens for the control-flow, plus a variable for each resource indicating their availability, and a variable for each object, indicating their existence. A function is enabled and can fire if it receives control (i.e. if at least a token marks its control-flow input arc), if at least all its *mandatory* roles are available and if at least all its *mandatory* input objects exist. The state of roles and objects is evaluated directly or via the respective range connectors. During a function's execution, the roles associated become unavailable and once the execution is concluded, the output objects are created (i.e. they become existent), and those ones that are indicated as *consumed*, are destroyed. Initial objects, i.e. those ones that are used by a function that follows a start event (e.g. the picture cut), exist before the process execution starts. A function does not wait for an optional role to become available. However, if such a role is available before the function is executed, it is treated as a mandatory role.

Based on the above transition rule, we can calculate the reachability graph for an iEPC to verify the behavioral correctness. For more details on the semantics of the control-flow, we refer to [14].

5 Exploring Integrated Process Configuration

We extend the iEPC meta-model to capture the variability of an integrated process model, by identifying variation points (*configurable nodes*) in the process model. A configuration assigns *configuration values* to each node, as well as constraints to restrict the combination of the allowed values. By configuring values to each nodes, we can derive a correct iEPC model from a starting Configurable iEPC (C-iEPC).

Variation points, represented via the use of a thick border in Fig. 1, can be any nodes of type function, role, object and connector. The values allowed for each variation point subsume the behavior of the non-configurable node, so that the configuration is carried out by restricting the process behavior. Accordingly, *Configurable functions* can be restricted from 'activated' (*ON*) to 'excluded' (*OFF*) or to 'optional' (*OPT*). The second option removes the function from the process model; the third permits deferring this choice till run-time, so that the decision to execute the function is made on an instance-by-instance basis. In the example, Music design is configurable: it can be set to *OFF* if the Director has planned not to have any music in the project, or to *OPT* to let the Director decide whether to have it or not, after the shooting phase.

Configurable control-flow connectors can be restricted to a less expressive connector type, or to a sequence of incoming control-flow nodes (in case of a join) or outgoing nodes (in case of a split). This is achieved by removing the connector altogether. An *OR* can be set to a regular *XOR*, *AND* or to a sequence. An *XOR* can only be set

to a sequence. An *AND* cannot be restricted. For instance, the configurable *XOR*-split (id. c_{14} in Fig. 1) can be set to the sequence starting with the event Design finished, to exclude the loop that reiterates the design phase after the Progress update. For further details on the configuration of the control-flow, we refer to [19].

Configurable roles and objects have two configuration dimensions: *optionality* and *specialization*, i.e. they can take a value for each dimension. The restriction of the attributes ‘optional’ and ‘mandatory’ of a role (object) falls into the first dimension. If a configurable role (object) is optional (*OPT*), it can be restricted to mandatory (*MND*), or it can be excluded from the process (*OFF*); if it is mandatory it can only be excluded. For example, the participation of the Composer and the production of Music cues can be excluded from the Spotting session, if the project does not feature music. The participation of the Producer in the Progress update can be made mandatory, or be left optional to defer the choice till run-time. Configurable roles and objects for which there exists a specialization in the hierarchy model, can be restricted to any of their specializations. As per the hierarchy model of Fig. 2, Picture cut can be specialized to Tape picture cut, or Producer to Co-Producer, should the Director need creative support for the Progress update. The availability of a specialization is depicted in the process model with a small pyramid in the role/object node. Since the editing process was conceived with the purpose of being configured, here we set each node for which a specialization was available, to the most general role (object) in the hierarchy.

Configurable input objects have a further configuration dimension – *usage*, such that those inputs that are ‘consumed’ (*CNS*) can be restricted to ‘used’ (*USE*). For instance, we can restrict Picture cut to used if its specialization is Tape, as an object of this type is never destroyed in the Picture editing.

Configurable range connectors have two configuration dimensions: *optionality* and *range restriction*. The same rules for roles and objects govern the possible changes of optionality values of a range connector. For example, the *OR*-join (c_{12}) can be made mandatory if the use of temp files is always required in the Progress update. The range restriction is achieved by increasing the lower bound and decreasing the upper bound, or a choice can be made for a single role (object) to be associated with the function linked to the connector, effectively removing the connector altogether. This is allowed if the lower bound is 1 and the node is in the connector’s preset (in case of a join), or in its postset (in case of a split). For example, the configurable connector ($2 : k$) (c_2) can be restricted to ($3 : k$) – all the supervisors have to partake in the Spotting session, or to ($2 : 2$) – exactly two of them have to partake. This is consistent with the configuration of the control-flow connectors, as the range connector subsumes all the control-flow connector types. Hence, an *OR* ($1 : k$) can be restricted to an *XOR* ($1 : 1$), to an *AND* ($k : k$), to a single node, but also to any other reduced range (e.g. $2 : k$). An *XOR* can only be restricted to a single node. An *AND* ($k : k$) cannot be restricted.

In Fig. 1, the object Music tracks is not configurable as output of Music design, as it is always produced by this function. On the contrary, the three objects for the sound tracks in output from Sound Design are configurable as we can choose which ones to produce, according to the input cues.

Under certain circumstances, it may not be allowed to freely set a configuration node, and this may depend on the configuration of other nodes. For example, there can

be dependencies between functions and roles, or objects and functions, determined by the domain, as shown in Section 3. *Configuration requirements* capture such restrictions in the form of logical predicates that govern the values of configurable nodes. These requirements can be classified according to the type of relation as follows (where M , S and U stand for the optionality, specialization and usage dimension, resp.):

Single Node requirements: constrain the configuration of a single node, where no dependency exists on other nodes. Req₁ (restriction of dimensions): the Picture cut associated to the Spotting session can only be specialized, but not restricted along its optionality dimension (i.e. it cannot be excluded as this the initial input to the process). Req₂: the Editor in Sound design cannot be specialized to Video Editor and the Editor in Picture editing cannot be specialized to Sound Editor, as the competencies are different. Req₃: the control-flow connector *XOR* (c_{14}) cannot be set to the sequence starting with the event Changes required, as this would lead to skip the whole premixing phase.

Connector–Connector requirements: constrain the configuration of two or more connectors. Req₄ (among role and object connectors): the range restriction of the *OR*-join (c_{11}) and of the *OR*-join (c_{12}) must be the same, so that at run-time, the choice of the role(s) linked by the first connector is consistent with the choice of the object(s) linked by the second connector, for the Progress update.

Function–Function requirements: constrain the configuration of two or more functions. Req₅: an editing project must have at least Music design or Sound design, thus we cannot exclude both. Req₆: Music premixing requires Music design as Sound premixing requires Sound design.

Role–Role requirements: constrain the configuration of two or more roles. Req₇ (on S): the Producer in Music premixing must be specialized in the same way as the Producer in Sound premixing, since these two roles are covered by the same person(s). Req₈ (on M): a Spotting session needs at least a Composer or a Sound Designer.⁴

Object–Object requirements: constrain the configuration of two or more objects. Req₉ (on S): the Picture cuts, the Edited pictures and the Deliverable must have the same specialization, to ensure a consistent propagation of the picture medium. Req₁₀ (on S, U): the Picture cut in Picture editing is consumed if and only if it is specialized to Film. Req₁₁ (on M): the exclusion of Dialog cues from Sound design implies the exclusion of Dialog tracks, since these are produced based on the cues.

Connector–Node requirements: constrain the configuration of connectors and configurable nodes. Req₁₂: the exclusion of function Progress update implies the restriction of the *XOR*-split (c_{14}) to the sequence starting with Design finished, as the repetition of the design phase depends on the result of the Progress update.

Function–Role requirements: constrain the configuration of functions and roles. Req₁₃ (on M): Music design must be excluded if the Composer is excluded from this function. On the other hand, if Sound Designer is excluded from Sound Design, this function can still be performed by the Editor.

Function–Object requirements: constrain the configuration of functions and objects. Req₁₄ (on M): Progress update cannot be excluded if Temp music file in Music design

⁴ The configuration of *OR*-join (c_1) allows one to restrict the choice of roles taken at run-time

or Temp sound file in Sound design are included, since the files are produced to be used later by this function. Otherwise, if Progress update is set to optional, the files cannot be made mandatory.

Role–Object requirements: constrain the configuration of roles and objects. Req₁₅ (on *M, S, U*): a Negcutter is only required if the project is edited and delivered on Film. Thus, if this role is mandatory, all the Picture cuts and Edited pictures, as well as the Deliverable, must be specialized to Film. In this case the Picture cut associated to the Picture editing cannot be set to used. Req₁₆ (on *M*): if a Composer does not partake in the Progress update, the Temp music file and the Music notes must be excluded as they are required by this role.

More complex requirements can be captured by combining requirements from the above classes. Fig. 3 shows the audio editing process that was followed by Bill Bennett to direct the feature film “Kiss or Kill”.⁵ This model is the result of configuring the reference process of Fig. 1 for an editing on Tape without music. Here, for instance, Music pre-mixing has been excluded and so has been Music design, as per Req₆. Moreover, since there is no Progress update, the control-flow loop for the repetition of the design phase has been removed, as per Req₁₂. The Editor for the Picture editing has been specialized to a Video Editor, as per Req₂. Since the editing is on Tape, the Picture cut in input to the Picture editing has been set to used and specialized to Tape, and the Negcutter has been excluded from this function, as per Req₁₅.

The algorithm to derive a correct iEPC from a configured C-iEPC is presented in the next section, which formalizes the notion of C-iEPC.

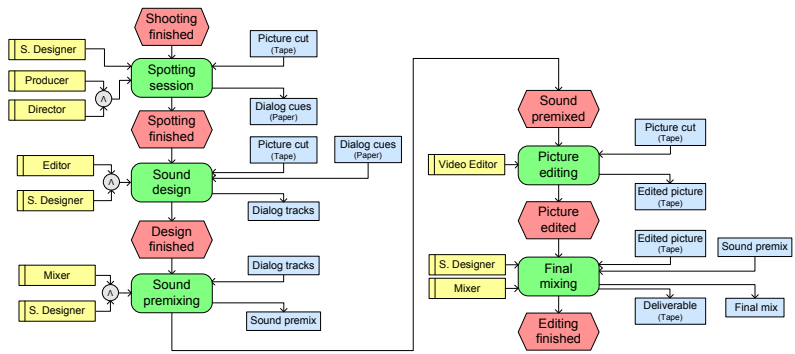


Fig. 3. The audio editing process model configured for a project without music

⁵ Kiss or Kill, 1997 (Australia), <http://www.imdb.com/title/tt0119467>

6 Formalizing Integrated Process Configuration

A C-iEPC is an extension of an iEPC where some nodes are identified as configurable, and a set of requirements is specified to constraint their values.

Definition 6 (Configurable iEPC). A configurable iEPC is a tuple C-iEPC = $(E, F_N, R_N, O_N, nm, C, A, L, F_N^c, R_N^c, O_N^c, C^c, RS^c)$, where:

- $E, F_N, R_N, O_N, nm, C, A, L$ refer to the elements of a syntactically correct iEPC,
- $F_N^c \subseteq F_N$ is the set of configurable functions,
- $R_N^c \subseteq R_N$ is the set of configurable roles,
- $O_N^c \subseteq O_N$ is the set of configurable objects,
- $C^c \subseteq C$ is the set of configurable connectors,
- RS^c is the set of configuration requirements.

All the auxiliary sets of Def. 4 are also defined for the configurable sets above. For example, $N^c = F_N^c \cup R_N^c \cup O_N^c \cup C^c$.

A configuration assigns values to each configurable node, according to the node type.

Definition 7 (Configuration). Let $M = \{MND, OPT, OFF\}$ be the set of optionality attributes, $U = \{USE, CNS\}$ the set of usage attributes, $CT = \{AND, OR, XOR\}$ the set of control-flow connector types and $CTS_{CF} = \{SEQ_n \mid n \in N_{CF}\}$ the set of sequence operators for control-flow. A configuration of C-iEPC is defined as $conf_{C-iEPC} = (conf_F, conf_R, conf_O, conf_C)$, where:

- $conf_F \in F_N^c \rightarrow \{ON, OPT, OFF\}$;
- $conf_R \in R_N^c \rightarrow M \times R$, (M is used for optionality and R for role specialization);
- $conf_O = conf_{IN} \cup conf_{OUT}$, where:
 - $conf_{IN} \in O_N^{IN^c} \rightarrow M \times O \times U$, (O is used for object specialization and U for usage);
 - $conf_{OUT} \in O_N^{OUT^c} \rightarrow M \times O$;
- $conf_C = conf_{C_{CF}} \cup conf_{C_R} \cup conf_{C_{IN}} \cup conf_{C_{OUT}}$, where:
 - $conf_{C_{CF}} \in C_{CF}^c \rightarrow CT \cup CTS_{CF}$, (CT is used for the connector's type and CTS_{CF} to configure the connector to a sequence of nodes);
 - $conf_{C_R} \in C_R^c \rightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup R_N)$, (\mathbb{N} and \mathbb{N} are used for lower bound increment and upper bound decrement, R_N is used to configure a role connector to a single role);
 - $conf_{C_{IN}} \in C_{IN}^c \rightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup O_N^{IN^c})$, ($O_N^{IN^c}$ is used to configure an input connector to a single input object);
 - $conf_{C_{OUT}} \in C_{OUT}^c \rightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup O_N^{OUT^c})$, ($O_N^{OUT^c}$ is used to configure an output connector to a single output object).

We define the following projections over the codomain of $conf_{C-iEPC}$:

Let $x \in R_N^c \cup O_N^{OUT^c}$, $\alpha \in \{R, OUT\}$ and $conf_\alpha(x) = (m, s)$, then $\pi^M(x) = m$ and $\pi^S(x) = s$. Let $x \in O_N^{IN^c}$ and $conf_{IN}(x) = (m, s, u)$, then $\pi^M(x) = m$, $\pi^S(x) = s$ and $\pi^U(x) = u$; Let $x \in C_R^c \cup C_{IN}^c \cup C_{OUT}^c$ and $\alpha \in \{R, IN, OUT\}$, then if $conf_{C_\alpha}(x) = (m, (p, q))$, then $\pi^M(x) = m$, $\pi^i(x) = p$ and $\pi^d(x) = q$, otherwise if $conf_{C_\alpha}(x) = (m, y)$, then $\pi^M(x) = m$ and $\pi^N(x) = y$.

The restrictions on the values each configurable node can take, are captured by the following partial orders. These are enforced in the definition of valid configuration.

Definition 8 (Partial Orders for Configuration). Let M, U, CT and CTS_{CF} as in Def. 7. The partial orders for configuration are defined as follows:

- $\preceq^M = \{MND, OFF\} \times \{MND\} \cup M \times \{OPT\}$ (on optionality),
- $\preceq^U = \{(n, n) \mid n \in U\} \cup \{(USE, CNS)\}$ (on usage),
- $\preceq^{CF} = \{(n, n) \mid n \in CT\} \cup \{XOR, AND\} \times \{OR\} \cup CTS_{CF} \times \{XOR, OR\}$ (on the type of control-flow connectors).

Definition 9 (Valid Configuration). A configuration $conf_{C-iEPC}$ is valid iff it fulfills the following requirements for any configurable node:

1. Roles and objects can be restricted to MND or OFF if they are OPT, or to OFF if they are MND ($\alpha \in \{R, O\}$):
 $\forall_{x \in R_N^C \cup O_N^C} [\pi^M(x) \preceq^M l_\alpha^M(x)].$
2. Roles and objects can be restricted to any of their specialization:
 $\forall_{x \in R_N^C \cup O_N^C} [\pi^S(x) \stackrel{\alpha}{\leftarrow} nm(x)].$
3. Input objects that are CNS can be restricted to USE:
 $\forall_{x \in O_{IN}^C} [\pi^U(x) \preceq^U l_O^U(x)].$
4. Control-flow OR connectors can be restricted to XOR, AND or to SEQ_n ; control-flow XOR connectors can be restricted to SEQ_n :
 $\forall_{x \in C_{CF}^C, n \in N_{CF}} [conf_{C_{CF}}(x) \preceq^{CF} l_C^T(x) \wedge (conf_{C_{CF}}(x) = SEQ_n \Rightarrow ((x \in C_{CF}^S \wedge (x, n) \in A_{CF}) \vee (x \in C_{CF}^J \wedge (n, x) \in A_{CF})))]$ (the sequence must be in the connector's postset in case of split or in its preset in case of join).
 Also, the configuration to SEQ_n must allow at least one path from a start to an end event: let $e_s \in E_s$ and $e_e \in E_e$, then
 $\exists_{p \in N_{CF}^+, p: e_s \mapsto e_e} \forall_{x \in C_{CF}^C \cap P(p)} [conf_{C_{CF}}(x) = SEQ_n \Rightarrow n \in P(p)].$
5. Range connectors can be restricted to MND or OFF if they are OPT, or to OFF if they are MND:
 $\forall_{x \in C_R^C \cup C_{IN}^C \cup C_{OUT}^C} [\pi^M(x) \preceq^M l_C^M(x)].$
6. Range connectors can be restricted to a smaller range or to a single node (role or object):
 - Range: $\forall_{x \in C_R^C \cup C_{IN}^C \cup C_{OUT}^C}$:
 - $\pi^i(x) = \pi^d(x) = 0$, if $lwb(x) = upb(x) = k$ (the AND case cannot be restricted),
 - $lwb(x) + \pi^i(x) \leq \begin{cases} upb(x) - \pi^d(x), & \text{if } upb(x) \in \mathbb{N}, \\ degree(x) - \pi^d(x), & \text{if } lwb(x) \in \mathbb{N} \text{ and } upb(x) = k; \end{cases}$
 - Node ($\alpha \in \{R, IN, OUT\}$):
 $\forall_{x \in C_R^C \cup C_{IN}^C \cup C_{OUT}^C} [\pi^C(x) = y \Rightarrow (link^\alpha(x, y) \wedge lwb(x) = 1)]$ (the node must be in the connector's postset in case of split or in its preset in case of join, and the lower bound must be 1).

Besides the structural requirements presented above, a valid configuration must fulfill the configuration requirements RS^C to be domain-compliant. We can express the configuration requirements of the editing process with the notation in Def. 7. We refer to the nodes by their id, as shown in Fig. 1. For example, Req_6 is $conf_F(f_5) = ON \Rightarrow conf_F(f_2) = ON$, Req_7 is $\pi^S(r_{15}) = \pi^S(r_{18})$ and Req_{10} is $\pi^U(o_{30}) = CNS \Leftrightarrow \pi^S(o_{30}) \stackrel{O}{\leftarrow} Film$.

The β_i -Algorithm maps a valid configuration and its C-iEPC into a syntactically correct iEPC, as stated by the theorem below. The general idea of this algorithm is to remove nodes as early as possible from the iEPC such that no unnecessary configurations are applied.

We first define some structural operators for an iEPC to be used in the algorithm.

Definition 10 (iEPC-Operators). *The following operators are defined for an iEPC:*

- *Remove-Operator* δ to delete the nodes in set X and their arcs:
 $\delta(iEPC, X)$ is an iEPC such for all sets $Y^\delta \in \delta(iEPC, X)$, $Y^\delta = Y \setminus X$, except $A^\delta = A \setminus \{(x, y) \in A \mid x \in X \vee y \in X\}$, and for all functions $\psi^\delta \in \delta(iEPC, X)$, $\psi^\delta = \psi|_{\text{dom}(\psi)^\delta}$.
- *Replace-Operator* ϱ to delete the nodes in set X and connect their preset and postset elements:
 $\varrho(iEPC, X) = \delta(iEPC, X)$ except $A^\varrho = A^\delta \cup \{(a, b) \mid \exists x \in X [a \in \overset{CF}{\bullet} x \wedge b \in x \overset{CF}{\bullet}]\}$.
- *Bypass-Operator* φ to insert two XOR connectors to bypass optional functions:
 Let $C_X = \{x^b \mid x \in X\} \cup \{x^a \mid x \in X\}$ be the set of new control-flow connectors that will be placed before and after the optional function. Then $\varphi(iEPC, X) = iEPC$ except $C^\varphi = C_{CF}^\varphi \cup C_R \cup C_{IN} \cup C_{OUT}$ where $C_{CF}^\varphi = C_{CF} \cup C_X$, and $A^\varphi = (A \setminus (N \times X \cup X \times N)) \cup \{(x^b, x) \mid x \in X\} \cup \{(x, x^a) \mid x \in X\} \cup \{(x^b, x^a) \mid x \in X\} \cup \{(y, x^b) \mid (y, x) \in A\} \cup \{(x^a, y) \mid (x, y) \in A\}$.
- *Events-Operator* Υ_E to remove sequences of events and add new arcs:
 Let $X = \{e \in E \mid \overset{CF}{\bullet} e \cap E \neq \emptyset\}$ be the set of events to be deleted and $A_X = \{(e, x) \in (E \setminus X) \times (N_{CF} \setminus X) \mid \exists e' \in X \exists p \in (X \cup \{e\})^+ [(e', x) \in A \wedge p : e \hookrightarrow e']\}$ be the set of arcs to be added. Then $\Upsilon_E(iEPC) = \delta(iEPC, X)$ except $A^{\Upsilon_E} = A^\delta \cup A_X$.
- *Connectors-Operator* Υ_C to remove sequences of control-flow connectors and add new arcs:
 Let $X = \{c \in C_{CF} \mid |\overset{CF}{\bullet} c| = |c \overset{CF}{\bullet}| = 1\}$ be the set of control-flow connectors to be removed, and let $A_X = \{(x, y) \in (N_{CF} \setminus X) \times (N_{CF} \setminus X) \mid \exists p \in (X \cup \{x, y\})^+ [p : x \hookrightarrow y]\}$ be the set of arcs to be added. Then $\Upsilon_C(iEPC) = \delta(iEPC, X)$ except $A^{\Upsilon_C} = A^\delta \cup A_X$.
- *Functions-Operator* Υ_F to add an event for any two consecutive functions:
 Let $E_X = \{e_{f,g} \mid (f, g) \in A \cap (F_N \times F_N)\}$ be the set of events to be added and $A_X = \{(f, e_{f,g}) \in F_N \times E_X\} \cup \{(e_{f,g}, g) \in E_X \times F_N\}$ be the set of arcs to be added. Then $\Upsilon_F(iEPC) = iEPC$ except $E^{\Upsilon_F} = E \cup E_X$ and $A^{\Upsilon_F} = (A \cup A_X) \setminus (F_N \times F_N)$.
- *Corona-Operator* Ω to identify roles, objects, and range connectors of functions:
 $\Omega(iEPC, X) = (\bullet X \cup X \bullet) \cup \bullet(\bullet X) \cup (X \bullet) \bullet \cap ((C \setminus C_{CF}) \cup R \cup O)$.

Definition 11 (β_i -Algorithm). *For a C-iEPC and conf_{C-iEPC} as one of its valid configurations, $\beta_i(C-iEPC, \text{conf}_{C-iEPC})$ defines an iEPC obtained as follows:*

1. *Populate all sets of $iEPC_1$ with the respective sets of C-iEPC.*
2. *Apply control-flow connector configuration and remove arcs not involving SEQ_n :*
 $iEPC_2 = iEPC_1$, except
 $l_{C,2}^T = l_{C,1}^T \oplus \{(c, \text{conf}_{CF}(c)) \mid c \in C_{CF}^C \wedge \text{conf}_{CF}(c) \in CT\}$ and
 $A_2 = A_1 \setminus (\{(c, n) \in C_{CF}^S \times c \overset{CF}{\bullet} \mid \exists_{n' \in c \overset{CF}{\bullet}, n' \neq n} [\text{conf}_{CF}(c) = SEQ_{n'}]\} \cup \{(n, c) \in \overset{CF}{\bullet} c \times C_{CF}^J \mid \exists_{n' \in \overset{CF}{\bullet} c, n' \neq n} [\text{conf}_{CF}(c) = SEQ_{n'}]\})$.⁶
3. *Remove nodes not on some path from an original start event to an original end event:*
 Let $e_s \in E_s$, $e_e \in E_e$ and let $N_X = \{n \in N_2 \mid \nexists_{p \in N_{CF}^+, p: e_s \hookrightarrow e_e} [n \in P(p)]\}$. Then
 $iEPC_3 = \delta(iEPC_2, N_X \cup \Omega(iEPC_2, N_X))$.
4. *Replace functions switched off with an arc, and remove their roles, objects and connectors:*
 Let $F_X = \{f \in F_{N,3} \mid \text{conf}_F(f) = OFF\}$. Then
 $iEPC_4 = \delta(\Upsilon_E(\varrho(iEPC_3, F_X)), \Omega(iEPC_3, F_X))$.

⁶ \oplus is the override operator

5. *Remove range connectors switched off, together with their roles and objects:*
 Let $C_X = \{c \in C_4 \setminus C_{CF,4} \mid \pi^M(c) = OFF\}$ and
 $RO_X = \{(R_{N,4} \cup O_{N,4}) \cap (\bullet C_X \cup C_X \bullet)\}$. Then
 $iEPC_5 = \delta(iEPC_4, C_X \cup RO_X)$.
6. *Remove roles and objects switched off:*
 $iEPC_6 = \delta(iEPC_5, \{ro \in R_{N,5} \cup O_{N,5} \mid \pi^M(ro) = OFF\})$.
7. *Remove range connectors no longer linked to roles and objects:*
 $iEPC_7 = \delta(iEPC_6, \{c \in C_6 \setminus C_{CF,6} \mid degree_6(c) = 0\})$.
8. *Replace all range connectors with a degree of one with arcs:*
 $iEPC_8 = \varrho(iEPC_7, \{c \in C_7 \setminus C_{CF,7} \mid degree_7(c) = 1\})$.
9. *Increment lower bound and decrement upper bound of configured range connectors:*
 $iEPC_9 = iEPC_8$ except
 $l_{C,9}^N = l_{C,8}^N \oplus \{(c, (lwb_8(c) + \pi^i(c), upb_8(c) - \pi^d(c))) \mid c \in C_8 \cap (C^c \setminus C_{CF}^c) \wedge upb_8 \neq k\} \cup \{(c, (lwb_8(c) + \pi^i(c), degree_8(c) - \pi^d(c))) \mid x \in C_8 \cap (C^c \setminus C_{CF}^c) \wedge lwb_8 \neq k \wedge upb_8 = k\}$.
10. *Align lower and upper bound of range connectors with potential change in degree:*
 $iEPC_{10} = iEPC_9$ except
 $l_{C,10}^N = l_{C,9}^N \oplus \{(c, (degree_9(c), upb_9(c))) \mid c \in C_9 \setminus C_{CF} \wedge lwb_9(c) > degree_9(c) \wedge (upb_9(c) \leq degree_9(c) \vee upb_9(c) = k)\} \cup \{(c, lwb_9(c), degree_9(c)) \mid c \in C_9 \setminus C_{CF} \wedge lwb_9(c) \leq degree_9(c) \wedge upb_9(c) > degree_9(c) \wedge upb_9(c) \neq k\} \cup \{(c, (degree_9(c), degree_9(c))) \mid c \in C_9 \setminus C_{CF} \wedge lwb_9(c) > degree_9(c) \wedge upb_9(c) > degree_9(c) \wedge upb_9(c) \neq k\}$.
11. *Apply configuration to roles, objects and range connectors:*
 Let $\alpha \in \{C, R, O\}$, then $iEPC_{11} = iEPC_{10}$ except
 $l_{\alpha,11}^M = l_{\alpha,10}^M \oplus \{(x, \pi^M(x)) \mid x \in N_{10} \cap (N^c \setminus N_{CF}^c)\}$,
 $l_{11}^U = l_{10}^U \oplus \{(x, \pi^U(x)) \mid x \in O_{N,10} \cap O_{N,10}^{IN^C}\}$, and
 $nm_{11} = nm_{10} \oplus \{(ro, \pi^S(ro)) \mid ro \in N_{10} \cap (R_N^c \cup O_N^c)\}$.
12. *Remove functions without mandatory role assignment:*
 Let $F_X = \{f \in F_{N,11} \mid \nexists_{r \in \bullet f} [l_R^M(r) = MND] \wedge \nexists_{c \in \bullet f} [l_C^M(c) = MND]\}$. Then
 $iEPC_{12} = \delta(\Upsilon_E(\varrho(iEPC_{11}, F_X)), \Omega(iEPC_{11}, F_X))$.
13. *Replace one-input-one-output connectors with arcs:*
 $iEPC_{13} = \Upsilon_E(\Upsilon_F(\Upsilon_C(iEPC_{12})))$.
14. *Insert connectors to bypass optional functions:*
 $iEPC_{14} = \varphi(iEPC_{13}, \{f \in F_{13} \mid conf_F(f) = OPT\})$.

Theorem 1. $\beta_i(C\text{-}iEPC, conf_{C\text{-}iEPC})$ is a syntactically correct $iEPC$.

Proof. We show that the properties of Def. 5 hold for $\beta_i(C\text{-}iEPC, conf_{C\text{-}iEPC})$ by discussing the different steps. We index intermediate syntax issues by $I_{s,r}$ where r refers to a syntax requirement and s to the step where it occurs.

1. The definition of $C\text{-}iEPC$ (Def. 6) implies that $iEPC_1$ is structurally correct.
2. Changing the connector labels according to the configuration values does not affect syntactical correctness. Removing non- SEQ_n arcs potentially results in nodes that are no more on a path from an original start event to an end event giving rise to issues ($I_{2,1}$). Furthermore, this step may lead to connectors with one input and one output arc ($I_{2,5}$).
3. This step resolves $I_{2,1}$ by removing all nodes that are not on a path from a start to an end node. Thanks to Def. 9, item 4, there must remain at least one path from a start to an end event, otherwise the configuration would not be valid. Furthermore, by removing also $\Omega(iEPC_2, N_X)$ all requirements related to roles and objects are fulfilled. Still, connectors may loose arcs and this may lead to connectors with one input and one output ($I_{3,5}$).

4. Since deleting the $\Omega(iEPC_3, F_x)$ nodes of all skipped functions and replacing them with an arc does not affect syntactical correctness, $iEPC_4$ inherits the issues $I_{2,5}$ and $I_{3,5}$ from $iEPC_3$. Possible event sequences are merged by the Υ_E operator. Furthermore, this step may add issues with connector cardinality if alternative branches are merged due to skipping functions ($I_{4,5}$).
5. Removing skipped range connectors and their role and object nodes may create problems with the requirement of a mandatory role ($I_{5,8}$).
6. Removing role and object nodes that are switched off potentially can create syntactical issues regarding range connector cardinality ($I_{6,5}$), mandatory roles of functions ($I_{6,8}$), and range connector bounds ($I_{6,10}$). Furthermore, $iEPC_6$ inherits issue $I_{2,5} - I_{4,5}$ and $I_{5,8}$.
7. This step resolves those cases related to $I_{6,5}$ in which the degree of the range connector is 0, but still there may be range connectors with a degree of 1.
8. This step resolves $I_{6,5}$. The resulting $iEPC_8$ inherits issues $I_{2,5} - I_{4,5}$, $I_{5,8}$, $I_{6,8}$, and $I_{6,10}$.
9. This step may resolve misalignment issues with upper bounds if, after the decrement, the new upper bound is equal or below the degree of a range connector. Still, misalignments of lower bounds cannot be resolved by this step. Therefore, the resulting $iEPC_9$ inherits issues $I_{2,5} - I_{4,5}$, $I_{5,8}$, $I_{6,8}$, and $I_{6,10}$.
10. Since the bounds are aligned with a potential change in degree, issue $I_{6,10}$ is resolved. $iEPC_{10}$ inherits issues $I_{2,5} - I_{4,5}$, $I_{5,8}$, and $I_{6,8}$.
11. This step does not affect syntactical correctness, since role and object nodes become mandatory which may resolve some issues with requirement 8.
12. This step resolves $I_{5,8}$ and $I_{6,8}$, but it may result in control-flow connectors with one input and out output, if alternative branches are merged due to functions switched off ($I_{12,5}$). Accordingly, $iEPC_{12}$ keeps issues $I_{2,5} - I_{4,5}$ and adds $I_{12,5}$.
13. This step resolves the remaining issues $I_{2,5} - I_{4,5}$ and $I_{12,5}$. Potentially created sequences of events are merged using Υ_E , and sequential functions are separated by a new event by Υ_F . Therefore, $iEPC_{13}$ is syntactically correct.
14. This step does not bring new issues. Therefore, the algorithm returns a syntactically correct $iEPC_{14}$.

7 Conclusion

This work has addressed a major shortcoming in existing configurable process notations: their lack of support for the data and resource perspectives. In doing so, we presented a rich meta-model for capturing role-task and object-task associations, that while embodied in the EPC notation, can be transposed to other notations. The study highlighted the intricacies that configurable process modeling across multiple perspectives brings. We identified interplays between perspectives. And while we define conditions to ensure syntactic correctness of individualized process models, we do not ensure semantic correctness. In future work, we will investigate techniques for preventing inconsistencies in the individualized process models, such as object flow dependencies that contradict control flow dependencies. While the proposal has been validated on a case study conducted with domain experts, further validation is required. This includes applying the proposal in other domains, but also designing tool support for configuration and conducting usability testing.

Acknowledgments. We thank Katherine Shortland and Mark Ward from the AFTRS for their valuable contribution to the design and validation of the reference models.

References

1. W. M. P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
2. W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
3. A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guizar, N. Kartha, C.K. Liu, R. Khalaf, D. Koenig, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. Committee specification 31 january 2007, OASIS, 2007.
4. J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, and D. Kuroпка. Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In *Proceedings of the 15th IRMA International Conference*, New Orleans, 2004. Gabler.
5. E. Bertino, E. Ferrari, and V. Atluri. The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. *ACM Transactions on Information and System Security (TISSEC)*, 2(1), February 1999.
6. T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
7. G. Engels, A. Förster, R. Heckel, and S. Thöne. Process Modeling Using UML. In M.Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede, editors, *Process-Aware Information Systems*, pages 85–117. Wiley, 2005.
8. D.F. Ferraiolo, R.S. Sandhu, S.I. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *Information and System Security*, 4(3):224–274, 2001.
9. F. Gottschalk, W.M.P. van der Aalst, and M.H. Jansen-Vullers. Configurable Process Models – A Foundational Approach. In *Reference Modeling. Efficient Information Systems Design Through Reuse of Information Models*, pages 59–78. Springer, 2007.
10. K. van Hee, O. Oanea, and N. Sidorova. Colored Petri Nets to Verify Extended Event-Driven Process Chains. In R. Meersman and Z. Tari, editors, *Proceedings of CoopIS/DOA/ODBASE 2005*, volume 3760 of *Lecture Notes in Computer Science*, pages 183–201. Springer-Verlag, 2005.
11. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
12. K. Jensen. *Coloured Petri nets: basic concepts, analysis methods and practical use: volume 1*. Springer-Verlag, 2nd edition edition, 1996.
13. J. Mendling, J. Recker, M. Rosemann, and Wil M.P. van der Aalst. Generating Correct EPCs from Configured CEPCs. In *Proceedings of the 21st Annual ACM Symposium on Applied Computing*, volume 2, pages 1505–1511, Dijon, France, 2006. ACM.
14. J. Mendling and W. M. P. van der Aalst. Formalization and Verification of EPCs with OR-Joins Based on State and Context. In *Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE'07)*, Norway, 11-15 June 2007.
15. M. zur Mühlen. Organizational Management in Workflow Applications Issues and Perspectives. *Information Technology and Management*, 5(3–4):271–291, July-October 2004.
16. J. Recker, J. Mendling, W.M.P. van der Aalst, and M. Rosemann. Model-Driven Enterprise Systems Configuration. In E. Dubois and K. Pohl, editors, *Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, volume 4001 of *Lecture Notes in Computer Science*, pages 369–383, Luxembourg, 2006. Springer.
17. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

18. M. La Rosa, J. Lux, S. Seidel, M. Dumas, and A.H.M. ter Hofstede. Questionnaire-driven Configuration of Reference Process Models. In J. Krogstie, A.L. Opdahl, and G. Sindre, editors, *Proceedings of the 19th Conference on Advanced Information Systems Engineering (CAiSE'07)*, volume 4495 of *Lecture Notes in Computer Science*, pages 424–438, Trondheim, Norway, 2007. Springer-Verlag.
19. M. Rosemann and W. M. P van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, 2007.
20. N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor and J. Falcao e Cunha, editors, *Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE'05)*, volume 3520 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, Berlin, 2005.
21. A.W. Scheer. *ARIS - Business Process Frameworks*. Springer, Berlin, 3rd edition, 1999.
22. S. Stephens. The Supply Chain Council and the SCOR Reference Model. *Supply Chain Management - An International Journal*, 1(1):9–13, 2001.
23. S.A. White et al. Business Process Modeling Notation (BPMN), Version 1.0, 2004.