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

M. La Rosa[1], M. Dumas[2,1], A.H.M. ter Hofstede[1], J. Mendling[1], and F. Gottschalk[3]

[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 [18] or the SAP Reference Model [5], 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 models* [15], 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 artifacts participating in the process.

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. We define a notion of valid configuration and an algorithm to individualize a configurable process model given a valid configuration. By construction, this algorithm ensures that the individualized process models are syntactically correct. 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 the modeling of object flow and resources in business processes and the notion of configurable process model. Section 3 introduces the working example and uses it to illustrate a meta-model that extends EPCs with resource and object flow modeling. Next, Section 4 explores the configuration of process models along the resource and object flow perspectives. Section 5 presents a formal model of a fully-featured C-EPC, which leads to the definition of an individualization algorithm. The paper concludes with a summary and an outlook on open issues.

2 Background and Related Work

2.1 Integrated Process Modeling

Business processes can be seen from a number of perspectives, including the control-flow, the data and the resource perspectives [9]. Control-flow is generally modeled in terms of activities and events related by control arcs and connectors. The resource perspective, on the other hand, is commonly modeled in terms of associations between activities and roles, where roles represent capabilities and/or organizational groups [1]. In UML Activity Diagrams (ADs) [6] and BPMN [19], this association is encoded by means of *swimlanes*. Each activity is associated with a swimlane representing a role or an organizational unit. UML ADs allow multiple swimlanes (or *partitions*) to be associated with an activity. In (extended) EPCs [17], symbols denoting roles or organizational units can be attached to tasks. In this paper, we define sophisticated role-based resource modeling features, which go beyond those found in UML ADs, BPMN and EPCs, and we layer configuration features on top of them. A notation-independent discussion of resource allocation for business processes is reported in [13, 16].

The flow of data and physical artifacts is generally captured by associating objects with activities. UML ADs support the association of object nodes with activity nodes to denote inputs and outputs. 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. Similar features are found in BPMN and extended EPCs. In this paper, we propose a more fine-grained approach to object flow modeling and mechanisms to capture variability in relation to tasks. Yet, we do not consider data mapping issues which are important for executable languages such as ADEPT$_{flex}$ [14], BPEL [3] or YAWL [2].

2.2 Configurable Process Modeling

Research on configurable business process models has focused on mechanisms for capturing variability along the control-flow perspective. Rosemann & van der Aalst [15]

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 set of process configuration operators based on skipping and blocking of tasks [7], and applied them to configure the control-flow of executable process modeling languages, such as YAWL and BPEL [8].

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, EPCs provide basic features for associating data and roles to tasks, which we extend in this paper. Finally, this choice allows us to build on top of the existing 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 reference process model on audio editing for screen post-production, which has been developed and validated in collaboration with subject-matter experts of the Australian Film Television & Radio School.[1] We chose this case study for the high level of creativity, and thus of variability, that characterizes the screen business. Indeed, the whole editing phase can radically change 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 roles 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 elements 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, in an integrated EPC (iEPC). A role, depicted on a function's left hand, captures a class of organizational resources that is able to perform that function: e.g. the role Producer captures the set of all the persons with this role in a given screen project. A role is dynamically bound to one concrete resource at run-time (e.g. the Producer associated with function *Spotting session* will be bound to Michelle Portland). A resource can be human or non-human (e.g. an information system or a robot), but for simplicity, we only consider human resources in the example. An object, depicted on a function's right hand, captures a physical or software artifact of an enterprise, that is used (input object) or produced (output object) by a function. Each object in the process model is statically bound to a concrete artifact.

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,

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

the OR-join between Composer and Sound Designer indicates that at least one of these roles is required to perform this activity. Composer is needed if the project features music, Sound Designer is needed if the project features sound, where sound consists of dialogs, effects (FX) and/or atmospheres (atmos). Based on the screening of the Picture cut, Composer and Sound Designer hold a Spotting session to decide what music or sound should be added at which point of time. This information is stored in the cues (e.g. Music cues for music). Picture cut is thus an input object, while the cues are output objects connected via an OR-split that indicates that at least one set of cues is produced, depending on the type of project. A spotting 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 bound and lower bound for a number of elements (roles or objects) that are required (where k refers to the indegree for a join or to the outdegree for a split; in this case $k = 3$).

Once the cues are ready, 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 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 one mandatory role is to be assigned to each function 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 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 Mixer and the Composer mix the Music tracks into a Music premix if the project has music, while the Mixer and the Sound Designer 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 ends with Final mixing, where the Mixer with the Sound Designer and/or the Composer release a Final mix using the available Premixes. A Deliverable may also be released by overlaying the premixes onto the Edited picture, should a demo of the video with the integrated audio be required.

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 with a role or object, by means of a specialization relation. Fig. 2 shows the hierarchy models for the

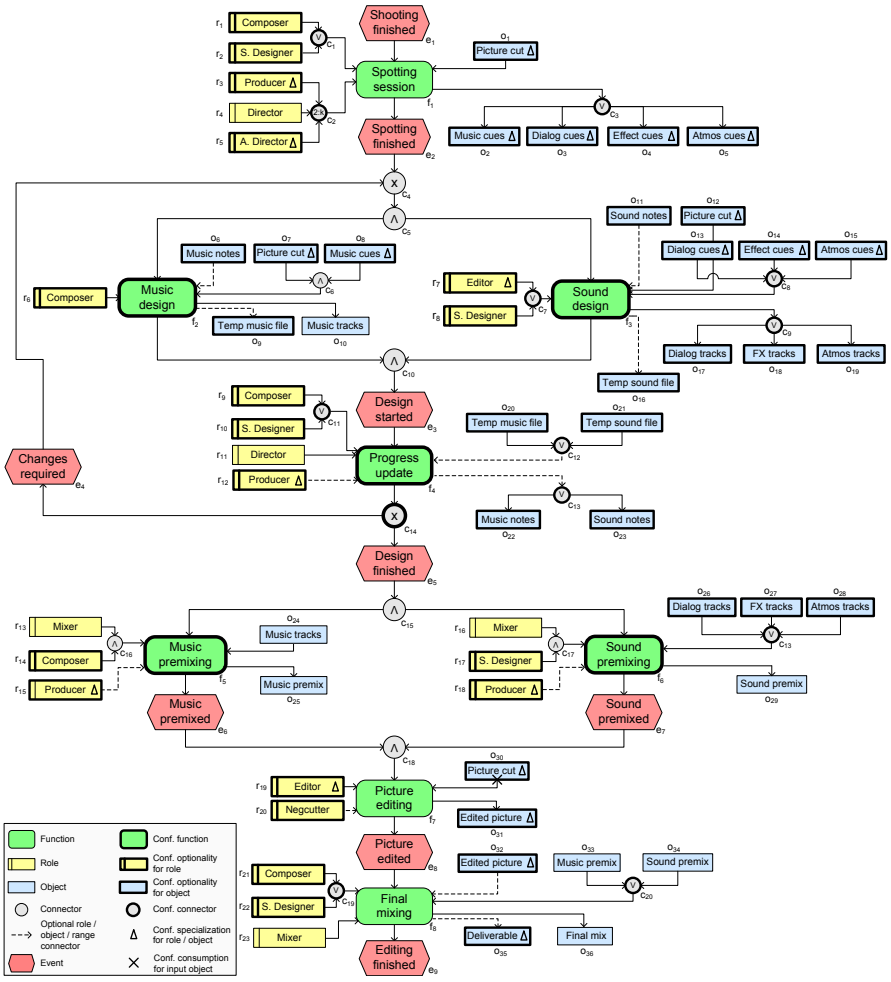

Fig. 1. The reference process model for audio editing

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). For an object, it represents a set of subtypes (e.g. 16mm and 35mm are two types of Film). The hierarchy models will be used later on in the configuration of the process model.

4 Exploring Integrated Process Configuration

A reference process model provides a generic solution that needs to be individualized to fit a specific setting. For this reason, process configuration can be interpreted as a restriction of the reference process model's behavior [7, 15]. Following this principle, we configure an integrated process model by restricting the behavior of a set of variation

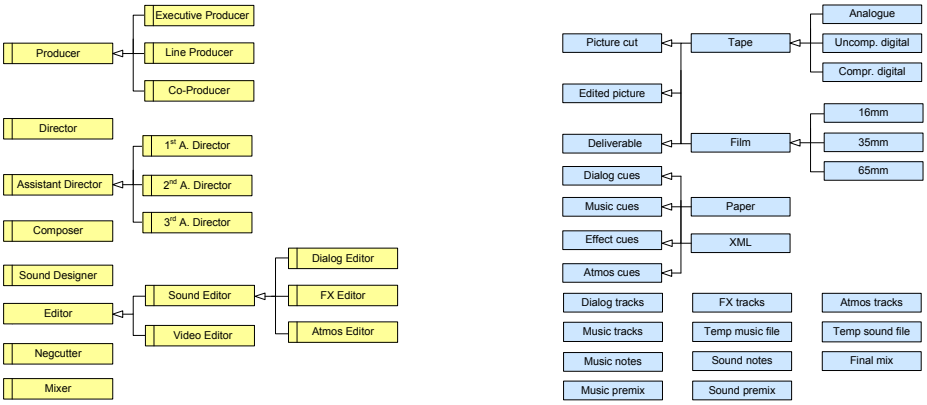

Fig. 2. The role-hierarchy model and the object-hierarchy model for the process of Fig. 1

points (*configurable nodes*) identified in the reference process model. A variation point can be any active node of a process model (function, role, object or connector), and is represented via the use of a thick border. A configuration assigns each variation point a (set of) *configuration value*(*s*) to keep or restrict the node's behavior. Since arcs and events are not active elements of a process model, they are not directly configurable. However, as a result of configuring a variation point, neighboring events and arcs can be affected (e.g. an event can be dropped). The extended iEPC meta-model that captures these concepts is called Configurable iEPC (C-iEPC). In the following, we describe the characteristics of each variation point.

Configurable functions can be left 'activated' (ON), or restricted to 'excluded' (OFF) or 'optional' (OPT). The second value removes the function from the process model (i.e. the function is skipped from the process flow). The third value permits deferring this choice until run-time, so that the decision whether to execute or skip 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, once the project has started.

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). The last option is achieved by removing the connector altogether. An OR can be restricted to an XOR, to an AND or to a sequence. An XOR can only be restricted to a sequence. An AND cannot be restricted. For instance, if the project cannot afford the repetition of music and sound design, due to the costs involved, the configurable XOR-split (id. c_{14} of the example, can be set to the sequence starting with event Design finished, so as to exclude the loop after function Progress update. For further details on the configuration of the control-flow, we refer to [15].

Configurable roles and *objects* have two configuration dimensions: *optionality* and *specialization*, i.e. they can take a value for each dimension. If a configurable role (object) is 'optional' (OPT), it can be restricted to 'mandatory' (MND), or to 'excluded' to be removed from the process (OFF); if it is 'mandatory' it can only be restricted to 'excluded'. For example, if a project does not feature music, the participation of the

Composer and the production of Music cues can be excluded from the Spotting session. 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, if the project does not support an editing on Film. Also, the Producer associated with Progress update can be specialized to Line Producer and made mandatory, should the Director need creative support in this phase. The availability of a specialization for a role or object, is depicted with a small pyramid in the node's right-hand side.

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 only a Picture cut on Film is destroyed during 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 optional OR-join connecting the temp files in Progress update, can be made mandatory if the temp files are always used by this function. The range restriction is achieved by increasing the lower bound and decreasing the upper bound, or a choice can be made for a single node (role or 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 range connector $(2 : k)$ associated with Spotting session, 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 any connector type. In fact, an OR is equivalent to a $(1 : k)$ range connector and 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.

Under certain circumstances, a configuration node may not be allowed to be freely set, and this may depend on the configuration of other nodes. In fact, there can be an interplay between the configuration of functions and roles, or objects and functions, determined by the domain in which the reference process model has been constructed. For example, an Edited picture is needed in Final mixing only if a Delivery is produced, otherwise it must be excluded. *Configuration requirements* capture such restrictions in the form of logical predicates that govern the values of configurable nodes. In the following, we classify these requirements according to the type of interplay and support this classification with examples taken from the model of Fig. 1 (where M, S and U stand for the optionality, specialization and usage dimension, resp.):

Single Node requirements: constrain the configuration of a single node, i.e. no dependency exists on other nodes. For example, the Picture cut associated with the Spotting session cannot be excluded as this is the initial input object to the whole process [Req_1]. Another example constraining the specialization of roles is given by the Editor in Sound design, which cannot be specialized to Video Editor, due to the capabilities required by associated the function [Req_2]. Concerning the control-flow, the XOR-split (c_{14}) after Progress update cannot be set to the sequence starting with the event Changes required only, as this would lead to skip the whole premixing phase [Req_3].

Connector–Connector requirements: constrain the configuration of multiple connectors. For instance, the two OR-joins for the roles and the input objects of Progress update (id. c_{11} and c_{12}) must be restricted the same way [Req$_4$]. The configuration of the former join allows the restriction of the run-time choice of which role is to partake in Progress update, while the configuration of the latter allows the restriction of which temp files to be used. Although the second connector is optional (i.e. no temp file may be used), a configuration where, e.g., the first OR is restricted to AND and the second one is restricted to a mandatory XOR must be denied. The reason is that if temp files are available, these need to be linked to the roles Composer and Sound Designer that will actually use them: the Composer will use the Temp music files, while the Sound Designer will use the Temp sound files.

Function–Function requirements: constrain the configuration of multiple functions. For example, an editing project deals with the editing of music and/or sound, so at least one function between Music design and Sound design must be present [Req$_5$]. Another constraint exists, e.g., between Music premixing and Music design, as the former needs to be dropped if the latter is excluded from the model [Req$_6$].

Role–Role requirements: constrain the configuration of multiple roles. For example, the Producer in Music premixing must be specialized in the same way as the Producer in Sound premixing, since these roles are typically covered by the same person [Req$_7$ (on S)]. To run a Spotting session, at least one role between Composer and Sound Designer need to be present [Req$_8$ (on M)].

Object–Object requirements: constrain the configuration of multiple objects. For instance, all the occurrences of the objects Picture cut and Edited picture must have the same specialization as the object Deliverable, to ensure a consistent propagation of the picture medium throughout the process [Req$_9$ (on S)]. The Picture cut in Picture editing is consumed if it is specialized to Film [Req$_{10}$ (on S, U)], as in this case the medium is physically cut (thus destroyed) and then spliced. Also, the exclusion of Dialog cues from Sound design implies the exclusion of Dialog tracks, since these are produced according to the cues [Req$_{11}$ (on M)].

Connector–Node requirements: constrain the configuration of connectors and nodes. For example, the exclusion of function Progress update implies the restriction of the XOR-split (c_{14}) to the sequence starting with Design finished, as at run-time the repetition of the design phase depends on the result of Progress update [Req$_{12}$].

Function–Role requirements: constrain the configuration of functions and roles. For instance, 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 [Req$_{13}$ (on M)].

Function–Object requirements: constrain the configuration of functions and objects. For example, function Progress update cannot be excluded if Temp music file in Music design or Temp sound file in Sound design are included, since the files are produced to be later used by this function. Otherwise, if Progress update is set to optional, the files cannot be made mandatory [Req$_{14}$ (on M)].

Role–Object requirements: constrain the configuration of roles and objects. An example is given by the role Negcutter, which is only required if the project is edited and delivered on Film. Thus, if this role is mandatory, all the occurrences of Picture cut and

Edited picture, and the Deliverable, must be specialized to Film. In this case the Picture cut in function Picture editing needs to be set to consumed [Req_{15} (on M, S, U)].

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".[2] This model is the result of configuring the reference process model of Fig. 1 for an editing on Tape without music. Here, for instance, Music premixing has been excluded and, as per Req_6, so has been Music design. Similarly, Progress update has been excluded, and thus, as per Req_{12}, the loop capturing the repetition of the design phase has also been removed. Moreover, the Editor in Picture editing has been specialized to a Video Editor (this complies with Req_2). Since the editing is on Tape, the Picture cut in input to Picture editing has been set to 'used' and specialized to Tape, and thus, as per Req_{15}, the Negcutter has been excluded from this function.

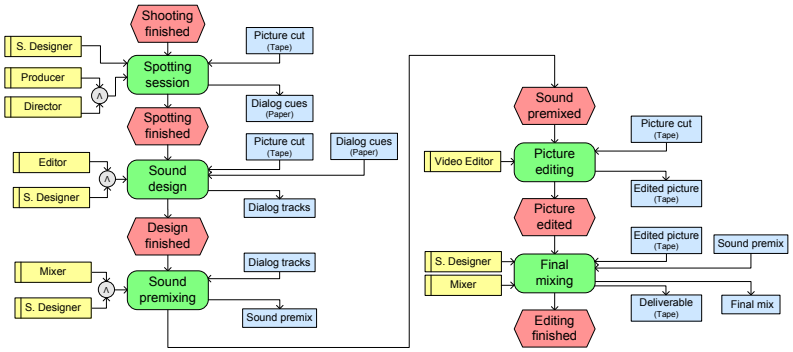

Fig. 3. The audio editing process model configured for a project without music

5 Correctness and Configuration of Integrated Process Models

This section presents an algorithm to generate an individualized iEPC from a C-iEPC with respect to a set of configuration values (i.e. a configuration). For example, this algorithm is able to generate the model shown in Fig. 3 from the model of Fig. 1 given a valid configuration. In doing so, the algorithm ensures the preservation of syntactic correctness, i.e. the individualized C-iEPC will be syntactically correct provided that the original C-iEPC was syntactically correct. To define this algorithm, we first need to define the notions of syntactically correct iEPC and valid configuration.

5.1 Integrated Business Process Model

In order to formally define the concepts of iEPC and correct iEPC, we first need to have a formal definition of role and object-hierarchies. A role (object) hierarchy is essentially a set of roles (objects) together with a specialization relation.

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

[2] Kiss or Kill, 1997 (Australia), `http://www.imdb.com/title/tt0119467`

- 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, acyclic*[3]*).*

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 definition of iEPC given below extends that of EPCs from [15], which focuses on the control-flow only. Specifically, iEPCs add a precise representation of roles and objects participating in the process. These roles and objects stem from the hierarchy-models defined above. 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 EPC formalizations, the range connector is not allowed in the control-flow, although a minimal effort would 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 node. 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,*
  - 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,*

[3] no cycles of length greater than one

  - $A_R \subseteq (R_N \times F_N) \cup (R_N \times C_R) \cup (C_R \times F_N)$ *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:*
  - $l_C^T \in C_{CF} \to \{AND, OR, XOR\}$ *specifies the type of control-flow connector,*
  - $l_C^N \in (C_R \cup C_{IN} \cup C_{OUT}) \to \mathbb{N} \times (\mathbb{N} \cup \{k\}) \cup \{(k,k)\}$*, specifies lower bound and upper bound of the range connector,*
  - $l_C^M \in (C_R \cup C_{IN} \cup C_{OUT}) \to \{MND, OPT\}$ *specifies if a role connector, an input connector or an output connector is mandatory or optional,*
  - $l_R^M \in R_N \to \{MND, OPT\}$ *specifies if a role node is mandatory or optional;*
  - $l_O^M \in O_N \to \{MND, OPT\}$ *specifies if an object node is mandatory or optional;*
  - $l_O^U \in O_N^{IN} \to \{USE, CNS\}$ *specifies if an input object node is used or consumed, where $O_N^{IN} = dom(A_{IN}) \cap O_N$.*

Given a connector c, let $l_C^N(c) = (n, m)$ for all $c \in C \setminus C_{CF}$. Then we use $lwb(c) = n$ and $upb(c) = m$ to 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.

We introduce the following notation 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 $O_N^{OUT} = dom(A_{OUT}) \cap O_N$;*
- $N = N_{CF} \cup N_R \cup N_{IN} \cup N_{OUT}$*, as its set of nodes;*
- $\forall_{n \in N_\alpha}\ {}^{\alpha}\!\bullet n = \{x \in N_\alpha \mid (x,n) \in A_\alpha\}$*, as the* α-preset *of n, $\alpha \in \{CF, R, IN, OUT\}$;*
- $\forall_{n \in N_\alpha}\ n\, \bullet\!^{\alpha} = \{x \in N_\alpha \mid (n,x) \in A_\alpha\}$*, as the* α-postset *of n, $\alpha \in \{CF, R, IN, OUT\}$;*
- $E_s = \{e \in E \mid |\ {}^{CF}\!\bullet\ e| = 0 \wedge |e\ {}^{CF}\!\bullet\ | = 1\}$ *as the set of start events;*
- $E_e = \{e \in E \mid |\ {}^{CF}\!\bullet\ e| = 1| \wedge |e\ {}^{CF}\!\bullet\ | = 0\}$ *as the set of end events;*
- $C_{CF}^S = \{c \in C_{CF} \mid |\ {}^{CF}\!\bullet\ c| = 1 \wedge |c\ {}^{CF}\!\bullet\ | > 1\}$ *as the set of control-flow split connectors;*
- $C_{CF}^J = \{c \in C_{CF} \mid |\ {}^{CF}\!\bullet\ c| > 1 \wedge |c\ {}^{CF}\!\bullet\ | = 1\}$ *as the set of control-flow join connectors;*
- $link^\alpha(x,y) = \begin{cases} (y,x) \in A_R\text{, if } \alpha = R\text{, returns the role arc from } y \text{ to } x, \\ (y,x) \in A_{IN}\text{, if } \alpha = IN\text{, returns the input arc from } y \text{ to } x, \\ (x,y) \in A_{OUT}\text{, if } \alpha = OUT\text{, returns the output arc from } x \text{ to } y; \end{cases}$
- $degree(x) = \begin{cases} |\ {}^{R}\!\bullet\ x|\text{, if } x \in C_R\text{, returns the indegree of a role connector,} \\ |\ {}^{IN}\!\bullet\ x|\text{, if } x \in C_{IN}\text{, returns the indegree of an input connector,} \\ |x\ {}^{OUT}\!\bullet\ |\text{, if } x \in C_{OUT}\text{, returns the outdegree of an output connector;} \end{cases}$
- $p = \langle n_1, n_2, \ldots, n_k \rangle$ *is a control-flow path such that $(n_i, n_{i+1}) \in A_{CF}$ for $1 \leqslant i \leqslant k-1$. For short, we indicate that p is a path from n_1 to n_k as $p : n_1 \hookrightarrow n_k$. Also, $P(p) = \{n_1, \ldots, n_k\}$ indicates the alphabet of p.*

We can now define a syntactically correct iEPC.

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:*
   $\forall_{e \in E} [| \overset{CF}{\bullet} e| \leqslant 1 \wedge |e \overset{CF}{\bullet} | \leqslant 1]$.
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:*
   $\forall_{f \in F_N} [\exists_{r \in \overset{R}{\bullet} f} [l^M_R(r) = MND] \vee \exists_{c \in \overset{R}{\bullet} f} [l^M_C(c) = MND]]$, *it follows that* $| \overset{R}{\bullet} f| > 0$.
9. *Roles and objects linked to connectors are mandatory:*
   $\forall_{r \in R_N} [r \in dom((R_N \times C_R) \cap A_R) \Rightarrow l^M_R(r) = MND]$,
   $\forall_{o \in O^{IN}_N} [o \in dom((O_N \times C_{IN}) \cap A_{IN}) \Rightarrow l^M_O(o) = MND]$,
   $\forall_{o \in O^{OUT}_N} [o \in dom((C_{OUT} \times O_N) \cap A_{OUT}) \Rightarrow l^M_O(o) = MND]$.
10. *Upper bound and lower bound of range connectors are restricted as follows:*
    $\forall_{c \in C_R \cup C_{IN} \cup C_{OUT}} [1 \leqslant lwb(c) \leqslant upb(c) \wedge (lwb(c) \leqslant degree(c) \vee upb(c) = k)]$,
    where $n \leqslant m$ *iff* $(n \leqslant m) \vee (m = k) \vee (n = m = k)$.

In the remainder, we assume an iEPC fulfills the above requirements. The editing process model of Fig. 1 is syntactically correct. However, Def. 5 does not prevent behavioral issues (e.g. deadlocks) that may occur at run-time. It is outside the scope of this paper to provide a formal definition of the dynamic behavior of iEPCs, as we only consider structural correctness in the context of configuration. Hence, here we briefly discuss its semantics for completeness, while for a formal definition we refer to a technical report [12].

The dynamic behavior of iEPC has to take into account the routing rules of the control-flow, the availability of the resources and the existence of the objects participating in the process. 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 role indicating the availability of the relative resource, and a variable for each object, indicating their existence. A function is enabled and can fire if it receives control, if at least all its *mandatory* roles are available and 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 associated roles 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 process objects, i.e. those ones that are used by a function that follows a start event (e.g. the Picture cut), exist before the 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.

5.2 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 constrain their values.

Definition 6 (Configurable iEPC). *A configurable $iEPC$ is a tuple $C\text{-}iEPC = (E, F_{_N}, R_{_N}, O_{_N}, nm, C, A, L, F^C_{_N}, R^C_{_N}, O^C_{_N}, C^C, RS^C)$, where:*

- $E, F_{_N}, R_{_N}, O_{_N}, nm, C, A, L$ *refer to the elements of a syntactically correct $iEPC$,*
- $F^C_{_N} \subseteq F_{_N}$ *is the set of* configurable functions*,*
- $R^C_{_N} \subseteq R_{_N}$ *is the set of* configurable roles*,*
- $O^C_{_N} \subseteq O_{_N}$ *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, $N^C = F^C_{_N} \cup R^C_{_N} \cup O^C_{_N} \cup C^C$. 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 $C\text{-}iEPC$ is defined as $conf_{_{C-iEPC}} = (conf_{_F}, conf_{_R}, conf_{_O}, conf_{_C})$, where:*

- $conf_{_F} \in F^C_{_N} \rightarrow \{ON, OPT, OFF\}$*;*
- $conf_{_R} \in R^C_{_N} \rightarrow M \times R$*, ($M$ is used for optionality and R for role specialization);*
- $conf_{_O} = conf_{_{IN}} \cup conf_{_{OUT}}$*, where:*
  - $conf_{_{IN}} \in O^{IN\,C}_{N} \rightarrow M \times O \times U$*, ($O$ is used for object specialization and U for usage);*
  - $conf_{_{OUT}} \in O^{OUT\,C}_{N} \rightarrow M \times O$*;*
- $conf_{_C} = conf_{_{C_{CF}}} \cup conf_{_{C_R}} \cup conf_{_{C_{IN}}} \cup conf_{_{C_{OUT}}}$*, where:*
  - $conf_{_{C_{CF}}} \in C^C_{CF} \rightarrow CT \cup CTS_{_{CF}}$*, ($CT$ is used for the connector's type and $CTS_{_{CF}}$ to configure the connector to a sequence of nodes);*
  - $conf_{_{C_R}} \in C^C_R \rightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup R_{_N})$*, ($\mathbb{N}$ 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);*
  - $conf_{_{C_{IN}}} \in C^C_{IN} \rightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup O^{IN}_N)$*, ($O^{IN}_N$ is used to configure an input connector to a single input object);*
  - $conf_{_{C_{OUT}}} \in C^C_{OUT} \rightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup O^{OUT}_N)$*, ($O^{OUT}_N$ 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 $x \in R^C_{_N} \cup O^{OUT\,C}_{N}$, $\alpha \in \{R, OUT\}$ and $conf_{_\alpha}(x) = (m, s)$, then $\pi^M(x) = m$ and $\pi^S(x) = s$. Let $x \in O^{IN\,C}_{N}$ and $conf_{_{IN}}(x) = (m, s, u)$, then $\pi^M(x) = m$, $\pi^S(x) = s$ and $\pi^U(x) = u$; Let $x \in C^C_R \cup C^C_{IN} \cup C^C_{OUT}$ 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, which are used in the definition of a valid configuration. For example, the partial order on the optionality dimension, prevents a 'mandatory' node from being configured to 'optional', while it allows the contrary.

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).*

With these elements, we are now ready to define the notion of *valid configuration*.

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:* $\forall_{x \in R_N^C \cup O_N^C} [\pi^S(x) \overset{\alpha}{\leftarrow} nm(x)]$.
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
   $\exists_{p \in N_{CF}^+, p: e_s \hookrightarrow e_e} \forall_{x \in C_{CF}^C \cap P(p)} [conf_{C_{CF}}(x) = SEQ_n \Rightarrow n \in P(p)]$.
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*),
   - $lwb(x) + \pi^i(x) \leqslant \begin{cases} upb(x) - \pi^d(x), \text{ if } upb(x) \in \mathbb{N}, \\ degree(x) - \pi^d(x), \text{ if } lwb(x) \in \mathbb{N} \text{ and } upb(x) = k; \end{cases}$

   • 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 be 1*).

Beside the structural requirements presented above, a configuration must fulfill the configuration requirements RS^C to be *domain-compliant*. We can express the configuration requirements of the editing process model using 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 $\pi^U(o_{30}) = CNS \Leftrightarrow \pi^S(o_{30}) \overset{O}{\leftarrow}$ Film.

The individualization algorithm applies a valid configuration to a syntactically correct C-iEPC, to generate a syntactically correct iEPC. The algorithm consists of a series of steps, each of which operates over a different type of element in a C-iEPC. The order of the steps in the algorithm has been chosen in such a way that no unnecessary operations are applied. For example, the control-flow connectors are configured first, as this operation may lead to skipping certain paths of the process model including connectors,

events and functions. Then, all the roles, objects and range connectors that are associated with functions no longer existing are removed as well. Finally, the remaining roles and objects are configured.

1. Apply control-flow connector configuration and remove arcs not involving sequences.
2. Remove nodes not on some path from an original start event to an original end event.
3. Replace functions switched off with an arc, and remove their roles, objects and connectors.
4. Remove range connectors switched off, together with their roles and objects.
5. Remove roles and objects switched off.
6. Remove range connectors no longer linked to roles and objects.
7. Replace all range connectors with a degree of one with arcs.
8. Increment lower bound and decrement upper bound of configured range connectors.
9. Align lower and upper bound of range connectors with potential change in degree.
10. Apply configuration of optionality dimension to roles, objects and range connectors; configuration of usage dimension to objects and configuration of specialization to roles and objects.
11. Remove functions without mandatory role assignment.
12. Replace one-input-one-output connectors with arcs.
13. Insert XOR-split, XOR-join and arcs to allow a bypass path for optional functions.

Fig. 4. Individualization algorithm.

The formal definition of this algorithm can be found in a technical report [10]. In the report, we also prove that any iEPC yielded by the algorithm fulfils the properties of syntactical correctness presented in Def. 5.

6 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. Also, while the proposal has been validated on a case study conducted with domain experts, further validation is required. The notion of configurable process model brings significant advantages, but concomitantly induces an overhead to the modeling lifecycle. In previous work [11] we have designed and implemented a tool, namely Quaestio, that provides a questionnaire-based interface to guide users through the individualization of configurable process models captured as C-EPCs. At present, we are extending this questionnaire-based framework to deal with C-iEPCs. The next step is to evaluate the framework by means of case studies in multiple domains and by conducting usability tests.

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 et al. Web Services Business Process Execution Language (WS-BPEL) ver. 2.0, Committee Specification, 31 Jan. 2007.
4. J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, and D. Kuropka. 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. T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.
6. 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.
7. 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.
8. F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *International Journal of Cooperative Information Systems*, 17(2):177–221, 2008.
9. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
10. M. La Rosa, M. Dumas, A. H. M. ter Hofstede, J. Mendling, and F. Gottschalk. Beyond Control-flow: Extending Business Process Configuration to Resources and Objects. 2007. Available at QUT ePrints, `http://eprints.qut.edu.au/archive/00011240`.
11. M. La Rosa, J. Lux, S. Seidel, M. Dumas, and A. H. M. ter Hofstede. Questionnaire-driven Configuration of Reference Process Models. In *Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 424–438, 2007.
12. J. Mendling, M. La Rosa, and A. H. M. ter Hofstede. Correctness of Business Process Models with Roles and Objects. 2008. Available at QUT ePrints, `http://eprints.qut.edu.au/archive/00013172`.
13. M. zur Mühlen. Organizational Management in Workflow Applications Issues and Perspectives. *Information Technology and Management*, 5(3–4):271–291, July-October 2004.
14. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
15. M. Rosemann and W. M. P van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, 2007.
16. 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)*, pages 216–232, 2005.
17. A.W. Scheer. *ARIS - Business Process Frameworks*. Springer, Berlin, 3rd edition, 1999.
18. S. Stephens. The Supply Chain Council and the SCOR Reference Model. *Supply Chain Management - An International Journal*, 1(1):9–13, 2001.
19. S.A. White et al. Business Process Modeling Notation (BPMN), Version 1.0, 2004.