Configurable Multi-Perspective Business Process Models

Marcello La Rosa [a,*] Marlon Dumas [b]
Arthur H.M. ter Hofstede [a] Jan Mendling [c]

[a] *Queensland University of Technology, GPO Box 2434, Brisbane 4001, Australia*
[b] *University of Tartu, J Liivi 2 Tartu 50409, Estonia*
[c] *Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany*

Abstract

A configurable process model provides a consolidated view of a family of business processes. It promotes the reuse of proven practices by providing analysts with a generic modeling artifact from which to derive individual process models. Unfortunately, the scope of existing notations for configurable process modeling is restricted, thus hindering their applicability. Specifically, these notations focus on capturing tasks and control-flow dependencies, neglecting equally important ingredients of business processes such as data and resources. This research fills this gap by proposing a configurable process modeling notation incorporating features for capturing resources, data and physical objects involved in the performance of tasks. The proposal has been implemented in a toolset that assists analysts during the configuration phase and guarantees the correctness of the resulting process models. The approach has been validated by means of a case study from the film industry.

Key words: business process, configurable process model, EPC

1 Introduction

Some business processes recur in similar forms from one company to another. For example, the term "order-to-cash" refers to a recurrent business process

* Corresponding author. Tel.: +61731389482; fax: +61731389390.
Email addresses: m.larosa@qut.edu.au (Marcello La Rosa), marlon.dumas@ut.ee (Marlon Dumas), a.terhofstede@qut.edu.au (Arthur H.M. ter Hofstede), jan.mendling@wiwi.hu-berlin.de (Jan Mendling).

that starts when a purchase order is received by a supplier and ends when the purchase order has been fulfilled and settled. Order-to-cash processes are found in a vast majority of companies. But while sharing common traits, order-to-cash processes in different companies or industry verticals may differ from one another in significant ways. An order-to-cash process for the delivery of goods is quite different from an order-to-cash process for services. In the first case, there is often a physical delivery that happens at a discrete point in time, and the condition of the goods can be checked upon receipt. On the other hand, the delivery of a service may occur over an extended period of time (e.g. 6 months) and may involve complex quality control activities.

Despite such differences, it would be inefficient if every time a company engages in modeling and re-designing its order-to-cash process, it did so "from scratch" without consideration of how other companies perform their order-to-cash process. Reference process models such as the Supply Chain Operations Reference (SCOR) model [41] or the SAP Reference Model [11], aim at enabling systematic reuse of proven practices across process (re-)design projects. They do so by capturing knowledge about common activities, information artifacts and flows encountered in specific application domains.

In general, reference process models take the form of libraries of process models structured as hierarchies. Analysts are expected to use these models and their associated documentation in order to derive process models for a specific set of requirements. In this way, reference process models provide an alternative to designing process models from scratch. However, reference process models in commercial use lack a representation of variation points and configuration decisions. In other words, they focus on the common traits found in recurrent business processes, but they do not capture possible variations in a systematic manner. As a result, analysts are given little guidance as to which model elements need to be removed, added or modified to meet a given set of requirements – a practice known as *individualization*.

The concept of *configurable process model* has been put forward as a means to address this limitation [37]. A configurable process model is a model that captures multiple variants of a business process in a consolidated manner. For example, Figure 1 shows a configurable process model (right-hand side) that consolidates two variants of a film shooting process: shooting on tape and shooting on film. The configurable model contains one variation point that can be configured into one of two variants. Although highly simplified, the example illustrates that a configurable process model captures the commonalities and the variability of a family of process models in a consolidated manner.

Given a configurable process model, analysts are able to define a *configuration* of this model by assigning values to its variation points based on a set of requirements. Once a configuration is defined, the model can be *individualized*

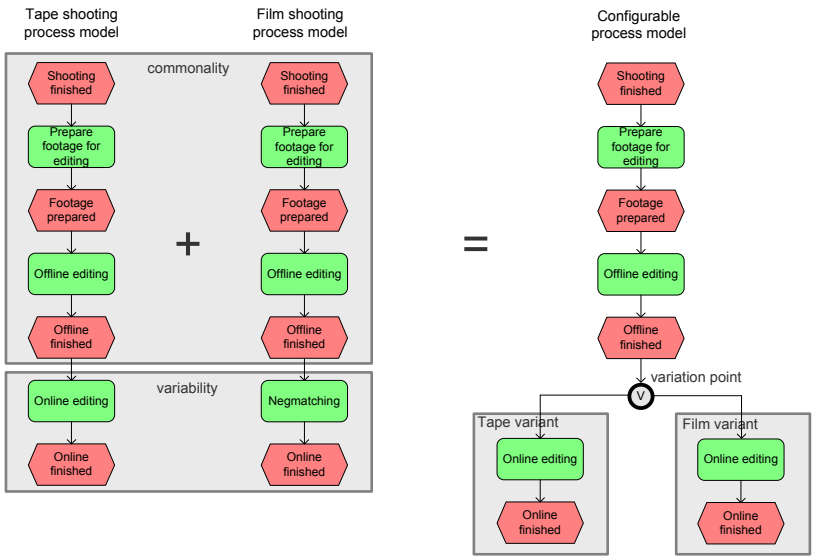

Fig. 1. A configurable process model consolidating two variants

automatically, thereby relieving analysts of this tedious and error-prone step.

Several notations for configurable process modeling have been proposed, including Configurable Event-driven Process Chains (C-EPCs) [37] and extensions of UML Activity Diagrams and BPMN as discussed later. The configurable process model shown in Figure 1 is represented using the C-EPC notation. This notation adds two concepts on top of the EPC notation: *configurable function* (a function that can be removed during the configuration phase) and *configurable connector* (a connector that can be removed or modified during the configuration phase). For example, Figure 1 features a configurable OR connector. During the configuration phase, either of the branches emanating from this connector can be removed. Alternatively, the configurable connector can be replaced by a non-configurable OR connector, or by an AND connector or by an XOR connector. This decision is made based on whether the modeler needs to configure the process to work with Tape only, with Film only, or with both, or to leave the option open until runtime.

It is customary in the literature to partition the elements found in process models into four major perspectives [19]. The *control-flow perspective* captures the occurrence and temporal ordering of activities. The *data perspective* (herein called *object perspective*) captures the data objects that are consumed and produced by the process and by its activities and how these data objects are used by decision points in the process. The *resource perspective* describes the organizational structure supporting the business process in the form of resources, roles, and groups. Finally, the *operational perspective* describes the elementary actions required to complete each activity, and how these actions map into underlying applications. Since the applications supporting a business process will almost always vary from one organization to another, common-

alities along the operational perspective are rare and thus this perspective is generally not found in reference process models. None of the reference models mentioned above covers the operational perspective, but they do contain elements across the other three perspectives. [1]

A major shortcoming of existing approaches to configurable process modeling is the lack of mechanisms for representing variability beyond the control-flow perspective. The situation is illustrated by Figure 1, which fails to capture the resources required during the performance of the editing and negmatching activities, and the inputs and outputs of these activities. This shortcoming limits the applicability of existing configurable process modeling notations.

The key contribution of this paper is an extension of the C-EPC notation with the notions of roles and objects. This extended notation, namely C-iEPC, supports a range of variations in the way roles and objects are associated to tasks. Given the subtle interplays between the control-flow, object and resource perspectives, decisions made along one of these perspectives may affect the other perspectives. Maintaining these perspectives synchronized is essential in order to ensure the correctness of the individualized process models. Accordingly, the paper also presents a notion of *valid configuration* and an algorithm to individualize a configurable process model with respect to a valid configuration in a way that guarantees the syntactic correctness of the individualized model. The proposal has been implemented in a toolset and has been applied to a case study in the film industry.

This paper is an extended and revised version of our previous work [23]. With respect to this previous work, this paper adds a formalization of the individualization algorithm, a theorem proving that the algorithm yields syntactically correct process models, a tool implementation and a case study.

The paper is structured as follows. Section 2 reviews previous work on multi-perspective and configurable process modeling. Section 3 introduces the integrated EPC (iEPC) notation, which allows one to capture business processes from the control-flow, object and resource perspectives in an integrated manner. Next, Section 4 presents the Configurable iEPC (C-iEPC) notation, allowing one to capture variation points on top of an iEPC. Section 5 presents a formal definition of C-iEPCs as well as the individualization algorithm. Section 6 describes the implemented toolset, while Section 7 presents the case study. Finally, Section 8 concludes the paper and discusses future work.

[1] Other perspectives include the *context perspective*, covering location and contextual attributes attached to activities, and the *performance perspective*, covering activity durations, costs, resource capacity, etc. These additional "non-functional" perspectives are outside the scope of this paper and may warrant a separate study.

2 Background and Related Work

We discuss related work on two aspects: *multi-perspective process modeling* and *configurable process modeling*. Given the high number of process modeling notations, it is impractical to be exhaustive. Accordingly, we focus on notations that emphasize activities and their dependencies, leaving aside process modeling notations based on data-flow, e.g. IDEF [29], and goal-based process modeling notations [21].

2.1 Integrated Multi-Perspective Process Modeling

The control-flow perspective of process models is commonly captured in terms of activities and events related by control-flow arcs and connectors. Such concepts can be found in virtually any process modeling notation. The resource perspective, on the other hand, is commonly modeled in terms of associations between activities and roles, where a role represents a set of capabilities and/or an organizational group [2]. In UML Activity Diagrams (ADs) [15] and BPMN [31], 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 (eEPCs) [39], symbols denoting roles or organizational units can be attached to functions. In this paper, we deal with role-based resource modeling features that go beyond these simple constructs found in UML ADs, BPMN and eEPCs. For example, whereas eEPCs allow one to express that a given resource *may* be needed for the performance of a function, or that a given object may be produced or consumed by a given function, the iEPC notation we introduce allows one to exactly state which combinations or resources may be used by an activity and which combinations of objects may be produced or consumed by an activity.

The flow of data and physical artifacts in a business process 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 for each of its output object nodes. Similar features are found in BPMN and extended EPCs. In this paper, we propose a more fine-grained object-activity associations and mechanisms to capture variability in relation to object-activity associations.

Languages for executable process modeling, such as ADEPT$_{flex}$ [34], BPEL [30] or YAWL [3], rely on global or net-level variables to capture data-flow between

actions. These languages support the definition of data mappings between global or net variables and task input and output parameters. Our aim is to define configurable process models for analysis and design. As such, our proposal does not cover aspects such as the definition of variables and data mappings, which are relevant at the execution level.

In addition to task-role associations, resource modeling in business processes also encompasses resource allocation and role-based access control. Russell et al. [38] identify a set of resource patterns describing various ways in which resources are represented and utilized in process models. Ferraiolo et al. [16] outline a reference model of well-accepted mechanisms for role-based access control, while Bertino et al. formalize role authorization constraints in workflow models [8]. As mentioned above, the models we deal with are analysis and design models. Accordingly, we do not deal with role-based access control and resource binding mechanisms which are relevant during the deployment and execution of business processes.

2.2 Configurable Process Modeling

Variability mechanisms have been widely studied in the field of Software Product Line Engineering (SPLE) [32]. Techniques developed in this field enable the configuration of software artifacts based on models that relate these artifacts to domain concepts (e.g. parameters, options or features). Among others, two research streams have emerged in SPLE, namely Software Configuration Management and Feature Diagrams. The former capture variability in terms of system dependencies, while the latter focus on the system features from a stakeholders perspective [4,13].

Existing methods for capturing variability in process models can be classified into four categories, based on their underlying variability mechanism: *configurable nodes*, *model projection*, *annotation* and *hiding and blocking of elements*.

Configurable nodes are used as a configuration mechanism in Configurable EPCs [37]. The key idea is that configuration options are associated with particular types of process model elements. We will discuss this approach in detail in Section 4.1.

Model projections are introduced as a variability mechanism in [5,6]. Since a reference process model typically contains information on multiple application scenarios, it is possible to create a projection for a specific scenario (e.g. a particular class of users) by fading out those process branches that are not relevant to the scenario in question. The configuration is conducted by setting *configuration parameters* defined in the form of simple attributes or logical

terms over characteristics. This can be elements of a model, but also elements of the meta-model. In this way, whole element types can be excluded. Those elements whose parameters evaluate to false are hidden.

[6] also introduces a distinction between *configuration* and *adaptation*. Configuration is concerned with the individualization of a reference process to fit the needs of an organization. Configuration does not lead to a fully-detailed process model because it is fundamentally limited by what is captured in the reference process model, which is generally a public and standardized model. Reference process models do not capture organization-specific details such as rules for calculating interests and penalties in a credit recovery process. After individualization, an adaptation step is required in order to incorporate such organization-specific details not covered in the reference process model. In this paper, we are concerned with configuration rather than adaptation.

Different approaches have been defined to achieve configuration by means of *annotations*. The PESOA (Process Family Engineering in Service-Oriented Applications) project [40] defines so-called *variant-rich process models* as process models extended with stereotype annotations to accommodate variability. These stereotypes are applied to both UML Activity Diagrams and BPMN models. The places of a process model where variability can occur are marked as variation points with the stereotype `<<VarPoint>>`. These can be activities in UML Activity Diagrams and tasks in BPMN. Further stereotypes including `<<Variant>>`, `<<Default>>`, `<<Abstract>>`, `<<Alternative>>`, `<<Null>>`, and `<<Optional>>` are used for the specification of different configuration options. A subset of these stereotypes proposed by the PESOA project appears in [33] where the authors focus on optionality and alternatives. These annotations can be applied to the control-flow and object-flow of UML Activity Diagrams. Comparable approaches to process model configuration based on annotations are introduced in [36,12,35].

In [17] the authors investigate a set of process configuration operators based on *hiding* (skipping) and *blocking* of model elements. These operators are applied to Labeled Transition Systems (LTSs). LTSs are a formal abstraction of computing processes, therefore any process model with a formal semantics (e.g. Petri Nets or YAWL) can be mapped to an LTS. The blocking operator corresponds to disabling the execution of an atomic action. In the LTS this means that a blocked edge cannot be taken anymore. Meanwhile, hiding corresponds to abstraction, i.e. the execution of an atomic action becomes unobservable. In the LTS a hidden edge is simply skipped, but the corresponding path is still taken. This approach has been subsequently applied to define a configurable extension of YAWL, namely C-YAWL [18].

Only two of these approaches take aspects beyond the configuration of control-flow into consideration. The approach by Becker et al. [5] can be used to hide

elements and element types in extended EPCs for configuration purposes, which includes non-control flow elements. 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). A second approach by Razavian et al. [33] is restricted to the configuration of simple forms of role-task and object-task associations that are present in UML ADs. As a result, both offer only basic features to configure resources and objects, such as removing a role or an object.

We conclude that richer mechanisms for the configuration of roles and objects in process models are needed. The two approaches that partially take these aspects into consideration suffer from limited expressiveness of the underlying meta-model. As a consequence, we will first formalize role and object modeling in a process model, leading to a notation we call integrated EPC (iEPC). We will then add features for capturing variability to this iEPC notation.

3 Integrated Business Process Modeling

We define our notation as an extension of EPCs. Three reasons underpin this choice. Firstly, EPCs are widely used by modelers and analysts for reference process modeling (cf. the SAP R/3 reference model). Hence, this can foster the adoption of the proposed extension. Secondly, eEPCs provide basic features for associating objects and resources 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 present our extensions in an abstract and formal manner to make them applicable beyond the scope of EPCs. In particular, the concepts we put forward can be directly transposed to other flowchart-like notations such as UML ADs and BPMN.

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. An integrated EPC (iEPC) extends an EPC by associating roles and objects to EPC functions. A role, shown to the left of a function, captures a class of organizational resources that is able to perform that function. For example, the role Producer associated with function Picture editing in Figure 2, captures the set of all the persons with this role in a given screen project. At run-time a role is dynamically bound to one concrete resource (e.g., Producer can be bound to Steven Spielberg).

Resources can be human or non-human, such as an information system or a robot. The SAP system associated with function Process Invoice in Figure 2 is an example of a non-human role. Therefore we can distinguish among manual

functions (those performed by human roles), automated functions (those performed by non-human roles) and semi-automated functions (those performed by both).

An object, shown to the right of a function, captures an information artifact (e.g. file) or a physical artifact (e.g. paper document or production material) of an enterprise that is used (input object) or produced (output object) by a function. For example, Edit notes is an input object for Picture editing in Figure 2, while Debit note is an output object for Process invoice. Each object in the process model is statically bound to a concrete artifact. Therefore if two objects in a model have the same label they are treated as being the same artifact.

Fig. 2. Two examples of task-role and task-object associations in iEPC.

The associations shown in Figure 2 are two examples of basic task-role and task-object associations. To illustrate more complex associations that can be captured in iEPC, we use the working example in Figure 3. This model is an exemplification of a reference process model on audio editing for screen post-production, which was developed and validated in collaboration with subject-matter experts of the Australian Film Television & Radio School (AFTRS) [2]. We chose this case study for the high level of creativity, and thus of variability, that characterizes the screen business. For example, the number of personnel involved depends greatly on the type of project. In an animation movie, dialogs are often recorded before the characters are finalized and scenes are complete. In a small budget feature film, the design and editing begins once the picture editing is complete, and is carried out by a single sound designer. On the other hand, on a high budget feature film the sound department can be made up of 30 or more roles. More information on the case study is provided in Section 7.

The first function of the model is Spotting session, which is performed once the shooting has been completed. Roles and objects can be linked to functions either directly or via a connector. A connector allows one to specify a logical condition for a set of roles or objects. For example, 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. Sound is a composition of dialogs, effects and/or atmospheres (atmos). Based on the screening of the Picture cut, Composer and Sound Designer hold a Spotting session to decide what music and sound should be added and at which point of time in

[2] `www.aftrs.edu.au`

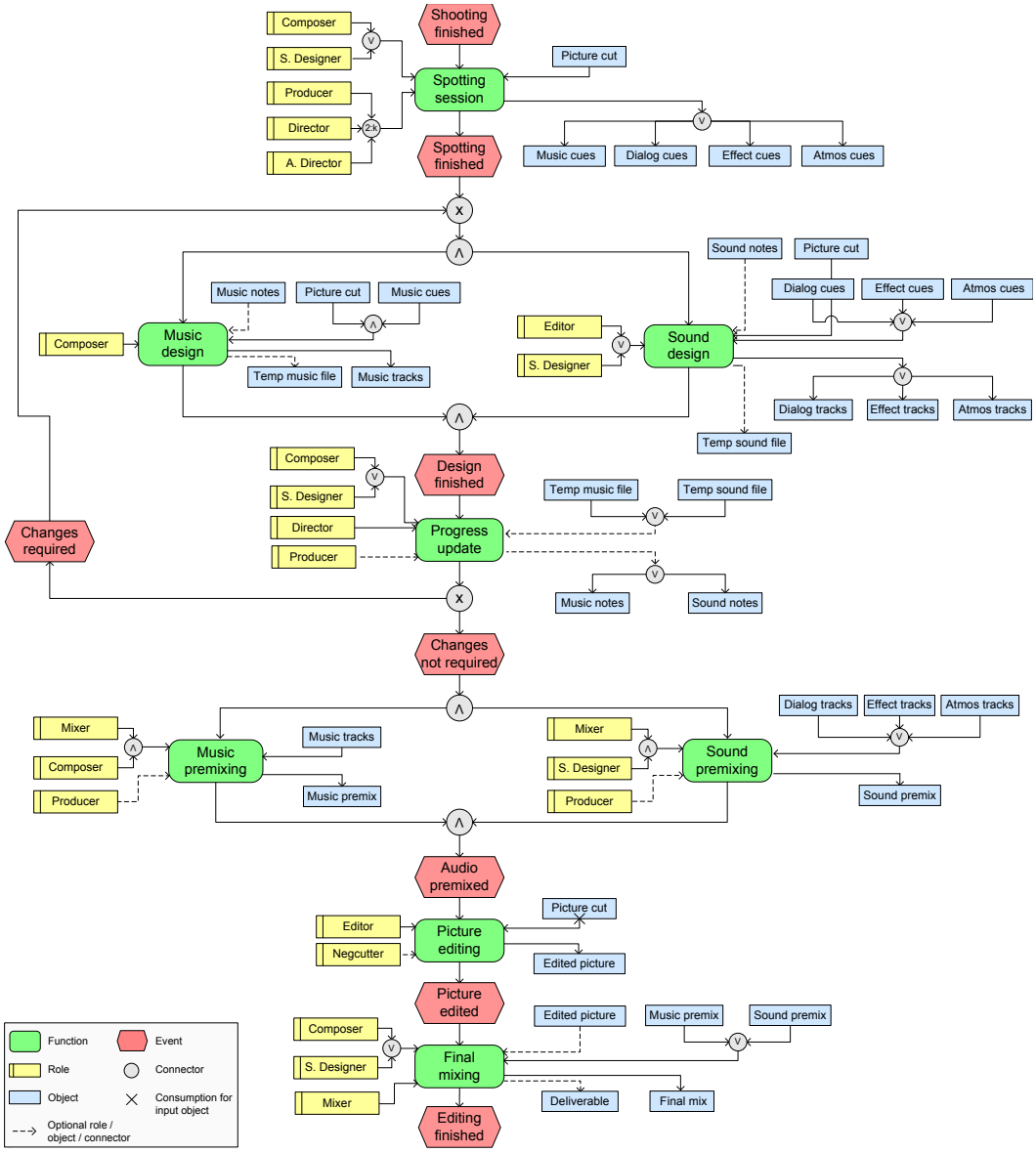

Fig. 3. Reference process model for audio editing at the AFTRS.

the Picture cut. This information is stored in the cues (e.g. Music cues for music). Picture cut is thus an input object while music cues, dialog cues, etc., are output objects. These are connected via an OR-split which imposes that at least one set of cues be produced as a result of the Spotting session. This choice depends on the type of project. For example, a documentary would typically have no effects.

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 lower bound and upper bound for the number of elements (roles or objects) that are required. The parameter k for a range connector refers to the

outdegree for a split or to the indegree for a join. In this case $k = 3$. A range connector subsumes the routing behavior of the common logical connectors of OR (equivalent to a range of $1 : k$), AND ($k : k$) and XOR ($1 : 1$). Therefore we consider all connectors involving roles and objects as being range connectors, although we maintain the standard EPC notation for OR ($\vee$), XOR ($\times$) and AND ($\wedge$). All the range values that can be captured through the range connector are illustrated in Figure 4 (the example of roles is shown). A function associated with more than one human role captures *teamwork*, i.e. a collaborative activity where each person contributes different skills. For example, function Spotting session captures the teamwork between the Composer, the Sound Designer and two roles among Producer, Director and Assistant Director.

Once the cues are ready the design of music and sound starts. In Music design, 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 or a role is optional, whereas a full arc indicates mandatoriness. The optionality of a group of roles/objects linked by a range connector is modeled by making the connector optional (see Figure 4). Sound design is usually more complex than Music design as it involves the recording of the Dialog, Effects 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. A cross below an object, like the cross below 'Picture cut' in Figure 3, indicates that the object is consumed by the function and is no longer available afterwards.

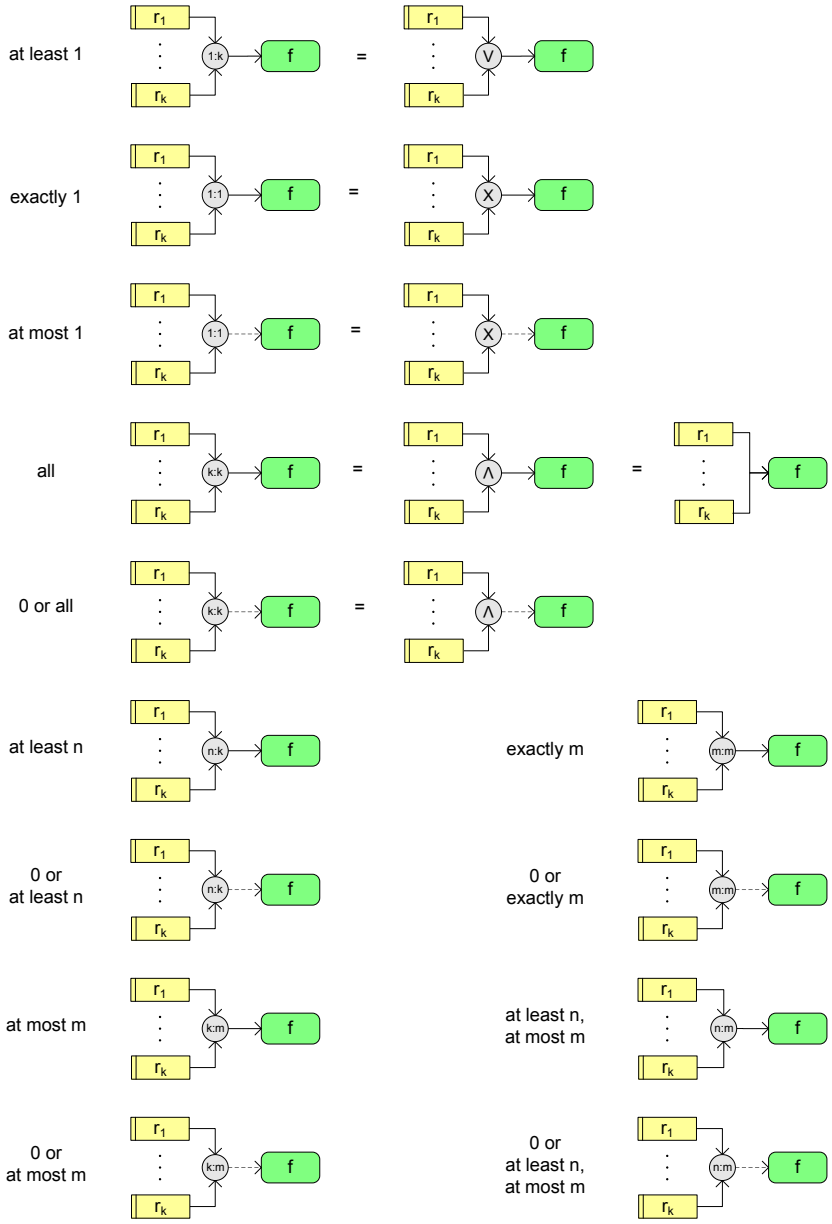

Fig. 4. Range values for the range connector.

The process ends with Final mixing, where the Mixer, helped by the Sound Designer and/or the Composer, releases 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.

Besides 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 audio editing process model there are five nodes labeled 'Producer' and four labeled 'Picture cut'. A hierarchy model also captures the specializations that can be associated with a role or an object, by means of a specialization rela-

tion. Figure 5 shows the hierarchy models for the roles and objects involved in the audio editing process, where the specialization relation is depicted by an empty arrow linking a specialized role (object) to its generalization. Typically, for a role this relation represents a separation of duties 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 formats (e.g. 16mm, 35mm and 65mm are three Film formats, while Paper and XML are two cues formats). The specializations in the hierarchy models will be used later on for configuration.

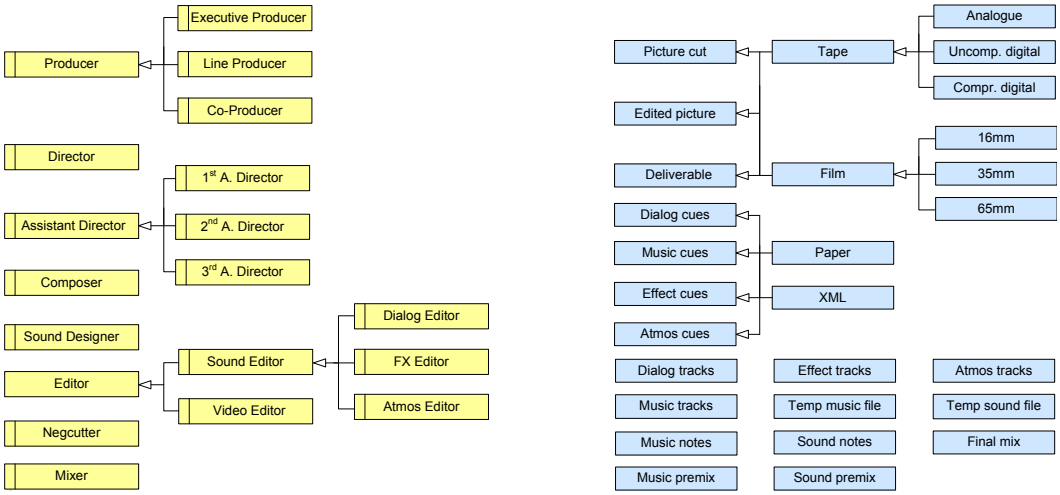

Fig. 5. The role-hierarchy model and the object-hierarchy model for the audio editing process model.

4 Configurable Integrated Process Models

In this section we extend iEPCs in order to capture variability and we discuss interplays among different process perspectives during configuration.

4.1 Configurable iEPCs

As mentioned in Section 1, a C-EPC is an EPC in which functions and connectors can be marked as "configurable". Given a C-EPC, a modeler can derive an individualized EPC by selecting a possible variant for each configurable element. Certain simple rules guide the space of possible variants for each configurable element. During configuration, configurable functions can be left *ON* or turned *OFF* (the function is replaced by an arc) or turned *OPT* (the decision whether to keep or discard the function is deferred until run-time). A configurable XOR connector with two outgoing branches can be configured

in three ways: either the left branch is removed, or the right branch is removed, or the connector is simply turned into a regular (non-configurable) XOR connector, meaning that the choice between the two branches is postponed until run-time. Finally, a configurable OR connector with two outgoing branches can be configured in five different ways as explained in the example in Section 1.

The Configurable iEPC (C-iEPC) notation that we propose in this paper extends C-EPCs by widening the spectrum of variation points beyond functions and control-flow connectors in order to include roles, objects and range connectors. With respect to C-EPCs, the C-iEPC notation adds three concepts: configurable roles, configurable objects and configurable range connectors. The configurable version of the reference process model for audio editing is shown as a C-iEPC in Figure 6, where variation points are indicated with a thicker border as in the C-EPC notation.

Configurable roles and *configurable objects* have two dimensions: *optionality* and *specialization*. If a configurable role (object) is 'optional' (*OPT*), it can be restricted to 'mandatory' (*MND*), or switched *OFF* to be removed from the process. If it is 'mandatory' it can only be switched *OFF*. 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 Figure 5, 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 top-right corner.

Configurable input objects have a further configuration dimension, namely *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. This is because a Picture cut is only physically destroyed if it is on Film.

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 for range connectors. 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 allows one to restrict the routing behavior of the connector at configuration time, i.e. before the actual execution of the process. This is achieved by increasing

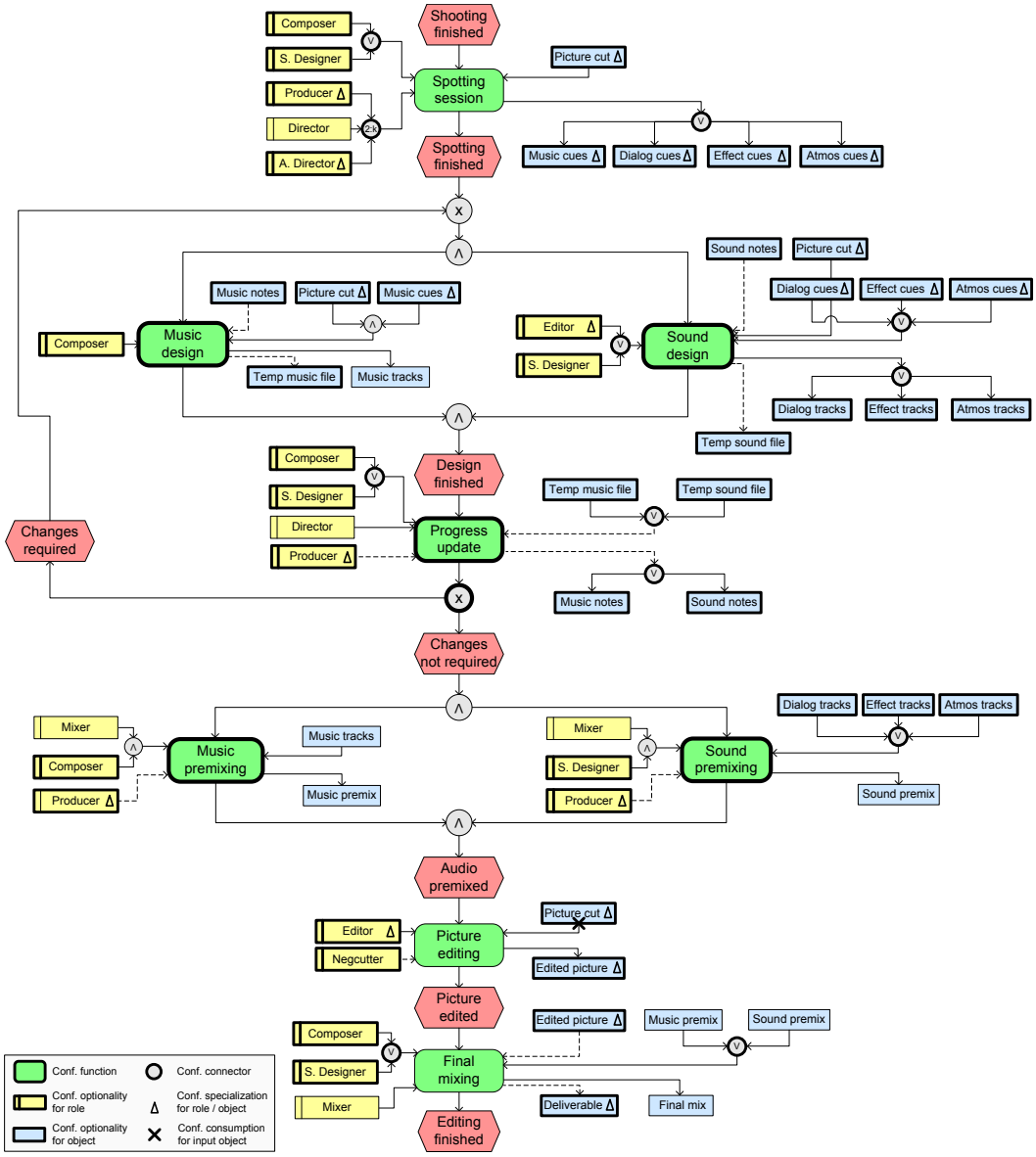

Fig. 6. The configurable version of the model in Figure 3.

the lower bound and/or decreasing the upper bound. Moreover, 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 latter option is similar to configuring a control-flow connector to a sequence of nodes, and is allowed if the lower bound is 1 and the node is in the connector's postset in case of a split, or in its preset in case of a join. 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 – but not to a single role.

The configuration of range connectors is consistent with the configuration of control-flow connectors, since as mentioned before the range connector sub-

sumes all connector types. In fact, a $1:k$ range connector is equivalent to an OR and can thus be restricted to an XOR $(1:1)$, to an AND $(k:k)$ and to a single node, but also to any other reduced range (e.g. $2:k$). A range $1:1$ can only be restricted to a single node while a rage $k:k$ cannot be restricted.

4.2 Configuration Interplays

Domain requirements may prevent a configuration node from being freely set, or more generally, may impose a set of configurable nodes to take only certain combinations of values. For example, in Figure 6, the role Editor associated with Sound design cannot be specialized to Video Editor since the capabilities required by the associated function include audio editing. Similarly, the Editor associated with Picture editing cannot be specialized to Sound Editor.

An example of a more intricate interdependency is that involving the role Negcutter. This role is required only if the project is edited and delivered on Film. Thus, if Negcutter is configured to *MND*, all the occurrences of objects Picture cut, Edited picture and Deliverable must be specialized to Film. Furthermore, in this case the Picture cut, which is an input object of Picture editing, must be restricted to *CNS* because it will be physically destroyed during the editing (the film roll is manually cut in a number of parts which are then spliced in a different order to obtain the desired editing).

Similarly, switching *OFF* function Progress update implies the restriction of the subsequent XOR-split to the sequence starting with event Changes not required. This is because at run-time the decision whether or not to repeat the design phase is determined by the outcome of Progress update.

Interdependencies may also involve range connectors. For instance, the two OR-joins for the roles and the input objects of Progress update must be configured the same way. The configuration of the first join allows the restriction of the run-time choice of which role is to partake in Progress update, while the configuration of the second join allows the restriction of which temp files have 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. This is because 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.

Besides domain requirements, structural requirements may also restrict the configuration space, in order to avoid the generation of incorrect individualizations. For example, we mentioned earlier that a function needs to have at least a mandatory role that can perform it. Thus, if a function is no longer as-

sociated with a role after configuration, it needs to be dropped from the model, because there are no resources that can execute it. This is the case for function Music design, which needs to be removed from the model if its sole role Composer is not available in the project. We discuss correctness requirements in Section 5, where we provide the definition of syntactically correct iEPC and formulate an algorithm that guarantees the model correctness during individualization. The integration of domain requirements with C-iEPC models is discussed in Section 6, where we describe a questionnaire-based approach for process model configuration and its tool support.

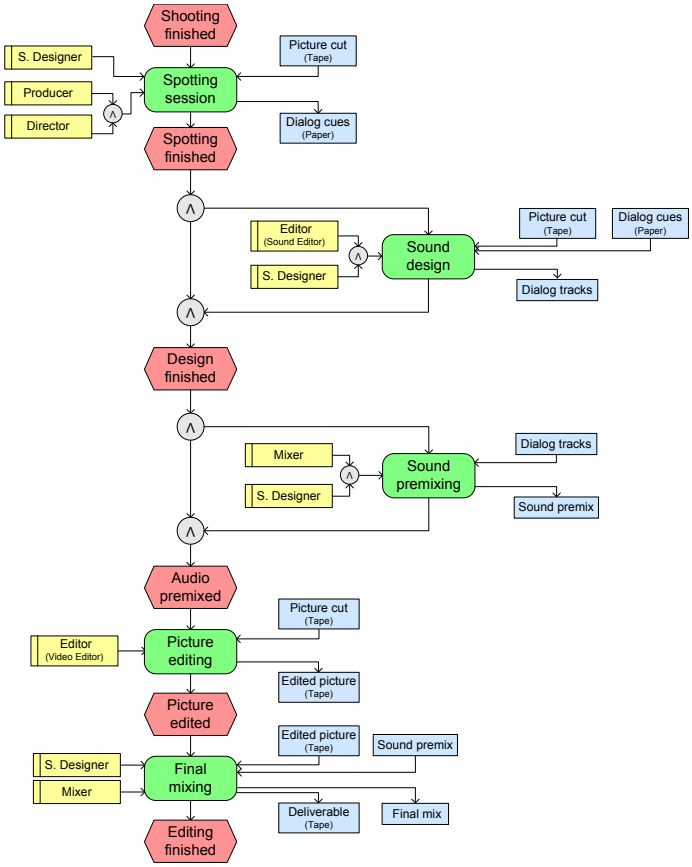

Fig. 7. The audio editing process model individualized for a project without music.

An example of a configured iEPC is depicted in Figure 7. This model describes the audio editing process that was followed by Bill Bennett to direct the feature film "Kiss or Kill" [7]. It is the result of configuring and individualizing the reference process model of Figure 6 for editing a feature movie without music on tape. Accordingly, functions Music design and Music premixing have been switched *OFF* and thus they have been replaced by an arc. Progress update has been excluded and thus the subsequent XOR-split has been configured so as to remove the possibility for iteration. The Editor in Picture editing has been specialized to Video Editor whereas the Editor in Sound design has been specialized to Sound editor. Furthermore, since the editing is on Tape, all instances of Picture cut, Edited picture and Deliverable have been specialized

to Tape, the Picture cut input to Picture editing has been set to 'used' and Negcutter has been switched *OFF*. This model complies with the domain and structural requirements described above.

5 Correctness and Configuration of Integrated Process Models

In this section we formalize the C-iEPC meta-model to provide a precise characterization of a C-iEPC configuration and discuss the requirements that need to be fulfilled to yield a syntactically correct individualized iEPC. First, we define the notion of iEPC and syntactically correct iEPC. Next, we provide the definition of C-iEPC and configuration and show an algorithm to individualize C-iEPCs. Finally, we prove that the algorithm guarantees the syntactic correctness of the models. For example, this algorithm is able to generate the model shown in Figure 7 from the model of Figure 6 given a configuration.

5.1 Integrated Business Process Model

The main iEPC elements and their relationships are shown in Figure 8 via an UML Class Diagram.

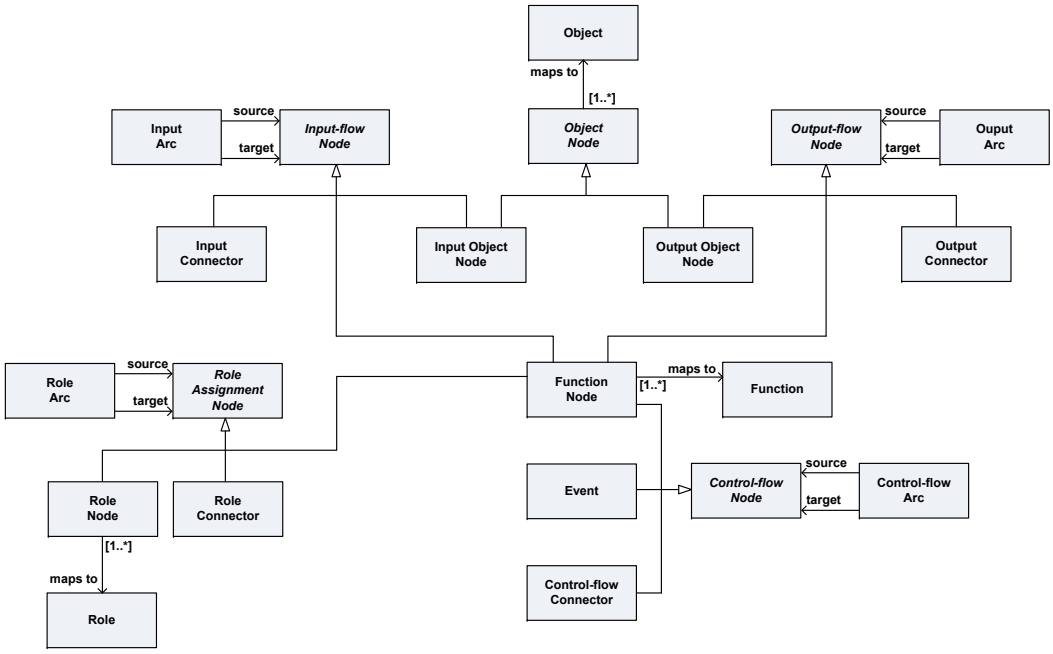

Fig. 8. UML Class Diagram showing the main iEPC elements (association cardinalities of 1 are omitted).

An iEPC is made up of control-flow nodes, role assignment nodes, input-flow nodes and output-flow nodes. Control-flow nodes divide into function

nodes (each mapped to one function), events and control-flow connectors. Role assignment nodes divide into role nodes (each mapped to one role), function nodes and role connectors. Finally, input-flow nodes divide into input object nodes, function nodes and input connectors, whereas output-flow nodes divide into output object nodes, function nodes and output connectors. Input and output object nodes are two types of object node and each object node is mapped to one object. A node can only be linked with another node of the same type, via the respective arc type. For example, a role node is linked with a role connector via a role arc. Links between nodes of different types are achieved through function nodes, which inherit from all four types of nodes and can thus be connected to all types of nodes.

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-hierarchy is essentially a set of roles with a specialization relation. Similarly, an object-hierarchy is a set of objects with a specialization relation.

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 and antisymmetric).*

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 information artifacts,*
- $\overset{O}{\leftarrow} \subseteq O \times O$ *is the* specialization *relation on* O *(*$\overset{O}{\leftarrow}$ *is transitive, reflexive and antisymmetric).*

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 EPC from [37] 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 function node, role node and object node represents an instance of a function, role or object, as illustrated in the UML diagram of Figure 8.

The range connector is modeled by a pair of natural numbers: lower bound (n) and upper bound (m). AND, OR and XOR correspond to a range connector respectively with $n = m = k$, with $n = 1, m = k$ and with $n = m = 1$. So we do not need to model the logical operators with separate connectors for roles

and objects, although they can be graphically represented with the traditional EPC notation, as in Figure 3. 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 Figure 3 as a dashed arc that links a node with a function, is modeled in iEPC as an attribute of the node being optional. Similarly, the consumption of input objects is modeled as an attribute of the object being consumed.

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 and Oh is a tuple $i\Upsilon_{F,Rh,Oh} = (E, F_N, R_N, O_N, nm, C, A, L)$, 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 object nodes (input connectors),*
  - C_{OUT} *is the set of range connectors for output object 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 op-*

tional,
- $l^{M}_{O} \in O_{N} \rightarrow \{MND, OPT\}$ *specifies if an object node is mandatory or optional,*
- $l^{U}_{O} \in O^{IN}_{N} \rightarrow \{USE, CNS\}$ *specifies if an input object node is used or consumed, where* $O^{IN}_{N} = dom(A_{IN}) \cap O_{N}$.

Given a connector c, let $l^{N}_{C}(c) = (n, m)$ for a $c \in C \setminus C_{CF}$. Then we use $lwb(c) = n$ and $upb(c) = m$ to refer to the lower bound and the upper bound of c. Moreover, if F, Rh and Oh are clear from the context, we drop the subscript from $i\Upsilon$. Also, we refer to function nodes, role nodes and object nodes simply as functions, roles and objects, wherever this does not lead to confusion.

Before defining a syntactically correct iEPC, we introduce the following subsets of nodes, functions and predicates to allow a more concise characterization of iEPCs.

Definition 4 (Auxiliary sets, functions and predicates) *Let F be a set of functions, Rh be a role-hierarchy model, Oh be an object-hierarchy model and $i\Upsilon = (E, F_{N}, R_{N}, O_{N}, nm, C, A, L)$ be an iEPC. Then:*

- $N_{CF} = E \cup F_{N} \cup C_{CF}$ *is the set of control-flow nodes;*
- $N_{R} = F_{N} \cup R_{N} \cup C_{R}$ *is the set of role assignment nodes;*
- $N_{IN} = F_{N} \cup O^{IN}_{N} \cup C_{IN}$ *is the set of input-flow nodes;*
- $N_{OUT} = F_{N} \cup O^{OUT}_{N} \cup C_{OUT}$ *is the set of output-flow nodes, where* $O^{OUT}_{N} = codom(A_{OUT}) \cap O_{N}$*;*
- $N = N_{CF} \cup N_{R} \cup N_{IN} \cup N_{OUT}$ *is the set of nodes;*
- $\forall_{n \in N_{\sigma}} \ {}^{\sigma}\bullet n = \{x \in N_{\sigma} \mid (x, n) \in A_{\sigma}\}$ *is the σ-preset of n, where $\sigma \in \{CF, R, IN, OUT\}$;*
- $\forall_{n \in N_{\sigma}} \ n \bullet^{\sigma} = \{x \in N_{\sigma} \mid (n, x) \in A_{\sigma}\}$ *is the σ-postset of n, where $\sigma \in \{CF, R, IN, OUT\}$;*
- $\bullet X = \bigcup_{x \in X, \sigma \in \{CF,R,IN,OUT\}} {}^{\sigma}\bullet x$ *is the union of the presets of X;*
- $X\bullet = \bigcup_{x \in X, \sigma \in \{CF,R,IN,OUT\}} x \bullet^{\sigma}$ *is the union of the postsets of X;*
- $E_{s} = \{e \in E \mid |\ {}^{CF}\bullet e| = 0 \wedge |e \bullet^{CF}| = 1\}$ *is the set of start events;*
- $E_{e} = \{e \in E \mid |\ {}^{CF}\bullet e| = 1| \wedge |e \bullet^{CF}| = 0\}$ *is the set of end events;*
- $C^{S}_{CF} = \{c \in C_{CF} \mid |\ {}^{CF}\bullet c| = 1 \wedge |c \bullet^{CF}| > 1\}$ *is the set of control-flow split connectors;*
- $C^{J}_{CF} = \{c \in C_{CF} \mid |\ {}^{CF}\bullet c| > 1 \wedge |c \bullet^{CF}| = 1\}$ *is the set of control-flow join connectors;*
- $link^{\sigma}(x, y) = \begin{cases} (y, x) \in A_{R}, \text{ if } \sigma = R, \text{ determines if } (y, x) \text{ is a role arc,} \\ (y, x) \in A_{IN}, \text{ if } \sigma = IN, \text{ determines if } (y, x) \text{ is an input arc,} \\ (x, y) \in A_{OUT}, \text{ if } \sigma = OUT, \text{ determines if } (x, y) \text{ is an output arc;} \end{cases}$

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

It follows that for all $f \in F_{N}$ $|f \overset{R}{\bullet}| = 0$, $|f \overset{IN}{\bullet}| = 0$ and $|\overset{OUT}{\bullet} f| = 0$; for all $r \in R_{N}$ $|\overset{R}{\bullet} r| = 0$; for all $o \in O_{N}$ $|\overset{IN}{\bullet} o| = 0$ and $|o \overset{OUT}{\bullet}| = 0$.

We can now define a syntactically correct iEPC. This definition extends that of syntactically correct EPC [37] by adding additional requirements for roles, objects and range connectors. For example, it specifies that range connectors associated with roles and input objects must be of type join, while range connectors associated with output objects must be of type split. Also, it imposes that functions be associated with at least one mandatory role or one mandatory role connector, and that roles and objects linked with range connectors be mandatory since the optionality of a group of roles/objects is modeled by making the connector optional.

Definition 5 (Syntactically Correct iEPC) *Let* F *be a set of functions,* Rh *be a role-hierarchy model,* Oh *be an object-hierarchy model and* $i\Upsilon_{F,Rh,Oh} = (E, F_{N}, R_{N}, O_{N}, nm, C, A, L)$ *be an iEPC.* $i\Upsilon$ *is syntactically correct if it fulfills the following requirements:*

(1) Every control-flow node is on a control-flow path from a start event to an end event: $\forall_{n \in N_{CF}} \exists_{e_s \in E_s, e_e \in E_e} [(e_S, n) \in A^{+}_{CF} \wedge (n, e_E) \in A^{+}_{CF}]$.

(2) There is at least one start event and one end event in $i\Upsilon$*:* $|E_s| > 0$ *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: $\forall_{f \in F_{N}} [|\overset{CF}{\bullet} f| = |f \overset{CF}{\bullet}| = 1]$.

(5a) 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),*

(5b) 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),*

(5c) 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),*

(5d) 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:*
$\forall_{r \in R_N} \ [r \in dom((R_N \times C_R) \cap A_R) \ \Rightarrow \ l_R^M(r) = MND]$,
$\forall_{o \in O_N^{IN}} \ [o \in dom((O_N \times C_{IN}) \cap A_{IN}) \ \Rightarrow \ l_O^M(o) = MND]$,
$\forall_{o \in O_N^{OUT}} \ [o \in dom((C_{OUT} \times O_N) \cap A_{OUT}) \ \Rightarrow \ l_O^M(o) = MND]$.

(10) *Upper bound and lower bound of range connectors are restricted as follows:* $\forall_{c \in C_R \cup C_{IN} \cup C_{OUT}} \ [1 \leq lwb(c) \leq upb(c) \ \wedge \ (lwb(c) \leq degree(c) \ \vee \ upb(c) = k)]$, *where* $n \leq m$ *iff* $(n \leq m) \ \vee \ (m = k) \ \vee \ (n = m = k)$.

The audio editing process model of Figure 3 is syntactically correct, and so is the individualized model of Figure 7.

The behavior of an 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 respective resource, and a variable for each object indicating its existence.

A function is enabled and can fire if it receives control, if resources are available for all its *mandatory* roles (i.e. persons in the case of manual functions) and if all its *mandatory* input objects exist. The state of roles and objects is evaluated directly or via the respective range connectors. Once a function terminates its execution, its output objects are created (i.e. they become existent), and those ones that are indicated as *consumed* are destroyed. Initial process objects, e.g. those ones that are used by a function that follows a start event, exist before the process execution starts. A function does not wait for a resource bound to an optional role to become available. However, if such a resource is available before the function is executed, it is treated as if its role was mandatory.

These definitions have implications for the correctness of a process model. Even if the control-flow of an iEPC is *sound*, i.e. the iEPC is syntactically correct and there are neither deadlocks nor livelocks (so-called *behavioral* issues), it is not guaranteed that the process can complete properly. The reason is that there can be further issues related to the object flow. Such issues may occur when, as a result of configuring a process model, a required input object does not exist at the time of executing a given function, despite the individualized model is syntactically correct. Below, we briefly illustrate two examples of how this issue can occur by using the audio editing process model of Figure 6.

The first example relates to the OR-join for input objects Temp music file and Temp sound file of function Progress update, which is optional. Since both Temp music file and Temp sound file are optional output objects respectively of functions Music design and Sound design, these objects might not be created in some instances of these functions. If so, there would be an issue if the above OR-join was configured to a 'mandatory' AND-join, because function Progress update could not be executed and thus the process instance would deadlock. The second example relates to the OR-join among the input objects Dialog tracks, Effect tracks and Atmos tracks of function Sound premixing. If we configured this to an AND-join, there would be instances in which this function might not be executed, given that the OR-split among the same output objects in Sound design produces non-deterministic behavior.

It is outside the scope of this paper to formally address the semantic issues of an iEPC. For a formal definition and a corresponding verification approach, we refer to separate work [26].

5.2 Integrated Process Configuration

A C-iEPC is an extension of an iEPC where a subset of its functions, connectors, roles and objects is identified as configurable. As per a C-EPC, control-flow connectors of type AND are not configurable.

Definition 6 (Configurable iEPC) *A configurable $i\Upsilon$ is a tuple $i\Gamma = (E, F_N, R_N, O_N, nm, C, A, L, F_N^C, R_N^C, O_N^C, C^C)$, where:*

- *$E, F_N, R_N, O_N, nm, C, A, L$ refer to the elements of an $i\Upsilon$, they form the iEPC component of the C-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*,*
- *$C^C \subseteq (C \setminus C_{AND})$ is the set of* configurable connectors*, where C_{AND} is the set of AND connectors.*

All the auxiliary sets of Definition 4 are also defined for $i\Gamma$. For example, with $N^C = F_N^C \cup R_N^C \cup O_N^C \cup C^C$ we indicate the set of configurable nodes.

We define a *C-iEPC valuation* as an assignment of values to each configurable node according to the node type. This valuation is then constrained to achieve a configuration.

Definition 7 (C-iEPC Valuation) *Let $i\Gamma = (E, F_N, R_N, O_N, nm, C, A, L, F_N^C, R_N^C, O_N^C, C^C)$ be a C-iEPC. Let also $M = \{MND, OPT, OFF\}$ be the set of optionality attributes, $U = \{USE, CNS\}$ the set of usage attributes,*

$CT = \{OR, XOR\}$ the set of configurable control-flow connector types and $CTS_{CF} = \{SEQ_n \mid n \in N_{CF}\}$ the set of sequence operators for the control-flow. A valuation of $i\Gamma$ is defined as $\mathcal{C}_{i\Gamma} = (\mathcal{C}_F, \mathcal{C}_R, \mathcal{C}_O, \mathcal{C}_C)$, where:

- $\mathcal{C}_F \in F_N^C \nrightarrow \{ON, OPT, OFF\}$;
- $\mathcal{C}_R \in R_N^C \nrightarrow M \times R$, ($M$ *is used for optionality and* R *for role specialization*);
- $\mathcal{C}_O = \mathcal{C}_{IN} \cup \mathcal{C}_{OUT}$, *where:*
  - $\mathcal{C}_{IN} \in O_N^{IN^C} \nrightarrow M \times O \times U$, ($O$ *is used for object specialization and* U *for usage*);
  - $\mathcal{C}_{OUT} \in O_N^{OUT^C} \nrightarrow M \times O$;
- $\mathcal{C}_C = \mathcal{C}_{C_{CF}} \cup \mathcal{C}_{C_R} \cup \mathcal{C}_{C_{IN}} \cup \mathcal{C}_{C_{OUT}}$, *where:*
  - $\mathcal{C}_{C_{CF}} \in C_{CF}^C \nrightarrow CT \cup CTS_{CF}$, ($CT$ *is used for the connector's type and* CTS_{CF} *is used to configure the connector to a sequence of nodes*);
  - $\mathcal{C}_{C_R} \in C_R^C \nrightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup R_N)$, ($\mathbb{N}$ *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*);
  - $\mathcal{C}_{C_{IN}} \in C_{IN}^C \nrightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup O_N^{IN})$, ($O_N^{IN}$ *is used to configure an input connector to a single input object*);
  - $\mathcal{C}_{C_{OUT}} \in C_{OUT}^C \nrightarrow M \times ((\mathbb{N} \times \mathbb{N}) \cup O_N^{OUT})$, ($O_N^{OUT}$ *is used to configure an output connector to a single output object*).

We define the following projections over the codomains of the component sets of $\mathcal{C}_{i\Gamma}$, to address each configuration value in a more compact way.

Definition 8 (C-iEPC Projections) *Let $i\Gamma = (E, F_N, R_N, O_N, nm, C, A, L, F_N^C, R_N^C, O_N^C, C^C)$ be a C-iEPC and $\mathcal{C}_{i\Gamma} = (\mathcal{C}_F, \mathcal{C}_R, \mathcal{C}_O, \mathcal{C}_C)$ be a valuation of $i\Gamma$:*

- *let $x \in R_N^C \cup O_N^{OUT^C}$, $\sigma \in \{R, OUT\}$ and $\mathcal{C}_\sigma(x) = (m, s)$, then $\pi^M(x) = m$ and $\pi^S(x) = s$;*
- *let $x \in O_N^{IN^C}$ and $\mathcal{C}_{IN}(x) = (m, s, u)$, 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 $\sigma \in \{R, IN, OUT\}$. If $\mathcal{C}_{C_\sigma}(x) = (m, (p, q))$, then $\pi^M(x) = m$, $\pi^i(x) = p$ and $\pi^d(x) = q$, otherwise if $\mathcal{C}_{C_\sigma}(x) = (m, y)$, then $\pi^M(x) = m$ and $\pi^N(x) = y$.*

The restrictions on the values each configurable node can take are captured a set of partial orders. These are used to constrain the possible values a C-iEPC valuation can take in order to be a valid valuation, i.e. a configuration of C-iEPC. The partial order on the optionality dimension prevents a 'mandatory' node from being configured to 'optional' while it allows the contrary. Similarly, the partial order on the usage dimension prevents a 'used' input object from being configured as 'consumed' while it allows the contrary. The partial order on the type of control-flow connectors prevents an XOR connector from being configured as an OR or AND, while it allows all other combinations.

Definition 9 (C-iEPC Partial Orders) *Let M, U, CT and CTS_{CF} be as in Definition 7. The partial orders for the configuration of a C-iEPC 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 *C-iEPC configuration*, which formalizes the concepts presented in Section 4.1.

Definition 10 (C-iEPC Configuration) *Let $i\Gamma = (E, F_N, R_N, O_N, nm, C, A, L, F_N^C, R_N^C, O_N^C, C^C)$ be a C-iEPC and $\mathcal{C}_{i\Gamma}$ be a valuation of $i\Gamma$. Then $\mathcal{C}_{i\Gamma}$ is a configuration of $i\Gamma$ 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:
$\forall_{x \in R_N^C \cup O_N^C} [\pi^M(x) \preceq^M l_\sigma^M(x)]$, *where* $(\sigma \in \{R, O\})$.

(2) Roles and objects can be restricted to any of their specializations:
$\forall_{x \in R_N^C \cup O_N^C} [\pi^S(x) \overset{\sigma}{\leftarrow} nm(x)]$, *where* $(\sigma \in \{R, O\})$.

(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:
$\forall_{x \in C_{CF}^C, n \in N_{CF}} [\mathcal{C}_{C_{CF}}(x) \preceq^{CF} l_C^T(x) \wedge (\mathcal{C}_{C_{CF}}(x) = SEQ_n \Rightarrow ((x \in C_{CF}^S \wedge (x,n) \in A_{CF}) \vee (x \in C_{CF}^J \wedge (n,x) \in A_{CF})))]$ *(the sequence must be in the connector's postset in case of a split or in the connector's preset in case of a join).*

(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 (i.e. to one role or one 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 $(\sigma \in \{R, IN, OUT\})$:

$\forall_{x \in C^C_R \cup C^C_{IN} \cup C^C_{OUT}} [\pi^N(x) = y \Rightarrow (link^\sigma(x,y) \wedge lwb(x) = 1)]$ (*the node must be in the connector's postset in case of split or in the connector's preset in case of join, and the lower bound must be 1*).

The individualization algorithm applies a configuration to a C-iEPC to generate an iEPC. If the configuration is *partial*, i.e. a configuration value is only assigned to a subset of the configurable nodes in the C-iEPC, then the resulting net will still be a C-iEPC. In this case further configurations would need to be applied in order to obtain an iEPC.

Each step of the algorithm operates over a different type of element in a C-iEPC. The order of these steps 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, objects and range connectors are configured.

The algorithm also removes all functions not associated with a mandatory role or mandatory role connector, and aligns the range of connectors with possible changes in degree resulting from switching some role or object *OFF*. Figure 9 illustrates this situation for the join "$2 : k$" associated with Spotting session in Figure 6. Assuming we configure this connector to "$3 : k$" ($\pi^i(c) = 1$, $\pi^d(c) = 0$), if at least one of the two configurable roles linked to it is switched *OFF*, its range needs to be restricted. In order to guarantee syntactic correctness, this rule also applies to non-configurable range connectors. For example, the AND join associated with Sound premixing must be dropped if Sound Designer is switched *OFF*.

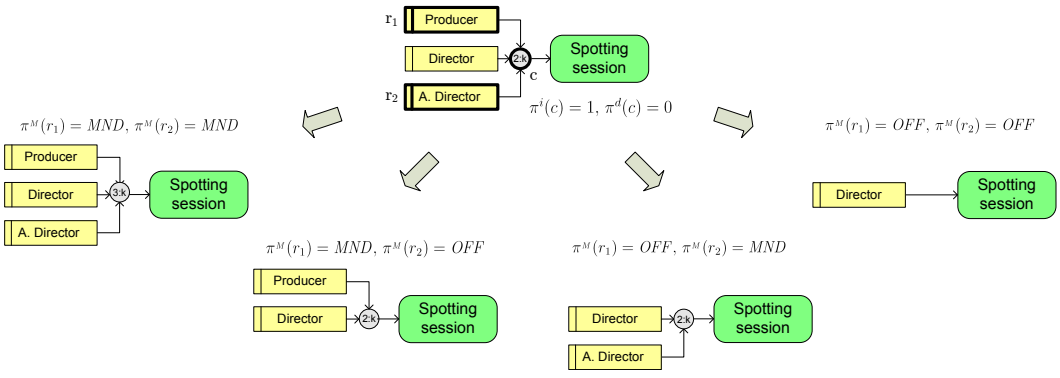

Fig. 9. The range of role and object connectors must be aligned with possible changes in the connector's degree.

When a function is switched *OFF*, the algorithm replaces the function with an arc, while when it is set to *OPT*, it bypasses the function by adding an XOR-split, an XOR-join and an arc in-between, as shown in Figure 10.

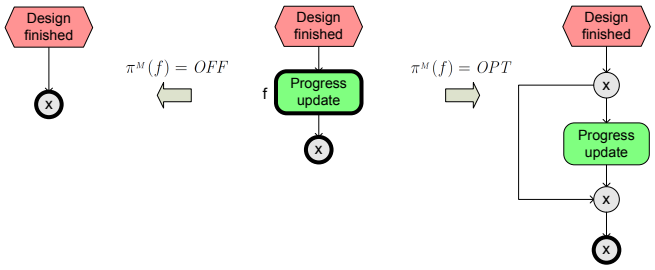

Fig. 10. The individualization of a function configured to *OFF* or *OPT*.

Before presenting the algorithm, we define some operators that allow us to perform the transformations that are needed to individualize a C-iEPC, e.g. removing a function or removing all the roles and objects of a function.

Definition 11 (iEPC Operators) *Let* $i\Gamma = (E, F_{N}, R_{N}, O_{N}, nm, C, A, L, F^{C}_{N}, R^{C}_{N}, O^{C}_{N}, C^{C})$ *be a C-iEPC. We define the following operators:*

- *Remove-Operator* δ *to delete the nodes in set* X *and their arcs:*
  $\delta(i\Upsilon, X)$ *is an iEPC such for all component sets* Y^{δ} *in* $\delta(i\Upsilon, X)$ *and all component sets* Y *in* $i\Upsilon$, $Y^{\delta} = Y \setminus X$, *except* $A^{\delta} = A \setminus \{(x,y) \in A \mid x \in X \vee y \in X\}$, *and for all functions* ψ^{δ} *that are components of* $\delta(i\Upsilon, X)$ *and all functions* ψ *that are components of* $i\Upsilon$, $\psi^{\delta} = \psi|_{dom(\psi^{\delta})}$.
- *Replace-Operator* ϱ *to delete the nodes in set* X *and connect their preset and postset elements:*
  $\varrho(i\Upsilon, X) = \delta(i\Upsilon, X)$ *except* $A^{\varrho} = A^{\delta} \cup \{(a,b) \in N_{CF} \setminus X \times N_{CF} \setminus X \mid \exists_{x,y \in X, \phi \in X^{+}, \phi: x \overset{*}{\hookrightarrow} y} [a \in \overset{CF}{\bullet} x \wedge b \in y \overset{CF}{\bullet}]\}$.
- *Bypass-Operator* φ *to insert two XOR connectors to bypass the functions in* X*:*
  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 functions in* X. *Then* $\varphi(i\Upsilon, X) = i\Upsilon$ *except* $C^{\varphi} = C^{\varphi}_{CF} \cup C_{R} \cup C_{IN} \cup C_{OUT}$ *where* $C^{\varphi}_{CF} = C_{CF} \cup C_{X}$, *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* Λ_{E} *to remove consecutive events and add new arcs:*
  Let $X = \{e \in E \mid \overset{CF}{\bullet} e \cap E \neq \varnothing\}$ *be the set of events to be deleted (i.e. those events that are preceded by an event) and* $A_{X} = \{(e,n) \in (E \setminus X) \times (N_{CF} \setminus X) \mid \exists_{e' \in X} \exists_{\phi \in (X \cup \{e\})^{+}} [(e',n) \in A \wedge \phi : e \hookrightarrow e']\}$ *be the set of arcs to be added. Then* $\Lambda_{E}(i\Upsilon) = \delta(i\Upsilon, X)$ *except* $A^{\Lambda_{E}} = A^{\delta} \cup A_{X}$.
- *Connectors-Operator* Λ_{C} *to replace single-entry-single-exit (sese) control-flow connectors with 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 (i.e. those connectors that are sese), and let* $A_{X} = \{(x,y) \in (N_{CF} \setminus X) \times (N_{CF} \setminus X) \mid \exists_{\phi \in (X \cup \{x,y\})^{+}} [\phi : x \hookrightarrow y]\}$ *be the set of arcs to be added. Then* $\Lambda_{C}(i\Upsilon) = \delta(i\Upsilon, X)$ *except* $A^{\Lambda_{C}} = A^{\delta} \cup A_{X}$.
- *Functions-Operator* Λ_{F} *to add a new event for any two consecutive func-*

tions:
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* $A_X = \{(f, e_{f,g}) \in F_N \times E_X\} \cup \{(e_{f,g}, g) \in E_X \times F_X\}$ *be the set of arcs to be added. Then* $\Lambda_F(i\Upsilon) = i\Upsilon$ *except* $E^{\Lambda_F} = E \cup E_X$ *and* $A^{\Lambda_F} = (A \cup A_X) \setminus (F_N \times F_N)$.

- *Corona-Operator* Ω *to identify the* corona *of functions, i.e. the roles, objects and range connectors associated with functions:*
  Let X *be the set of control-flow nodes for which a corona needs to be identified. Then* $\Omega(i\Upsilon, X) = ((\bullet X \cup X\bullet) \cup \bullet(\bullet X) \cup (X\bullet)\bullet) \cap ((C \setminus C_{CF}) \cup R \cup O)$.[3]

We can now present the individualization algorithm for C-iEPCs.

Definition 12 (Configured C-iEPC) *Let* $i\Gamma = (E, F_N, R_N, O_N, nm, C, A, L, F_N^C, R_N^C, O_N^C, C^C)$ *be a C-iEPC and* $\mathcal{C}_{i\Gamma}$ *be one of its configurations.* $\beta_{i\Gamma}(i\Gamma, \mathcal{C}_{i\Gamma})$ *defines a (C-)iEPC* $i\Psi$ *obtained as follows:*

(1) Populate all component sets of $i\Psi_1$ *with the respective sets of* $i\Gamma$.
(2) Apply control-flow connector configuration and remove arcs not involving SEQ_n*:*
$i\Psi_2 = i\Psi_1$, *except*
$l^T_{C,2} = l^T_{C,1} \oplus \{(c, \mathcal{C}_{CF}(c)) \mid c \in C^C_{CF} \wedge \mathcal{C}_{CF}(c) \in CT\}$ *and*[4]
$A_2 = A_1 \setminus (\{(c,n) \in C^S_{CF} \times c \overset{CF}{\bullet} \mid \exists_{n' \in c\overset{CF}{\bullet}, n' \neq n} [\mathcal{C}_{CF}(c) = SEQ_{n'}]\} \cup$
$\{(n,c) \in \overset{CF}{\bullet} c \times C^J_{CF} \mid \exists_{n' \in \overset{CF}{\bullet} c, n' \neq n} [\mathcal{C}_{CF}(c) = SEQ_{n'}]\})$.
(3) Remove nodes not on some path from an original start event to an original end event:
Let $N_X = \{n \in N_{CF,2} \mid \nexists_{\phi \in N^+_{CF}, e_s \in E_s, e_e \in E_e, \phi: e_s \hookrightarrow e_e} [n \in \alpha(\phi)]\}$. *Then* $i\Psi_3 = \delta(\delta(i\Psi_2, N_X), \Omega(i\Psi_2, N_X))$.
(4) Replace functions switched OFF with arcs and remove their coronas (this may result in consecutive events to be removed):
Let $F_X = \{f \in F_{N,3} \mid \mathcal{C}_F(f) = OFF\}$. *Then*
$i\Psi_4 = \delta(\Lambda_E(\varrho(i\Psi_3, F_X)), \Omega(i\Psi_3, F_X))$.
(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*
$i\Psi_5 = \delta(i\Psi_4, C_X \cup RO_X)$.
(6) Remove roles and objects switched OFF:
$i\Psi_6 = \delta(i\Psi_5, \{ro \in R_{N,5} \cup O_{N,5} \mid \pi^M(ro) = OFF\})$.
(7) Remove range connectors no longer linked to roles and objects:
$i\Psi_7 = \delta(i\Psi_6, \{c \in C_6 \setminus C_{CF,6} \mid degree_6(c) = 0\})$.
(8) Replace all range connectors with a degree of one with arcs:

[3] If $X \cap F = \varnothing$ then $\Omega(i\Upsilon, X) = \varnothing$.
[4] $\oplus$ is the override operator.

$i\Psi_8 = \varrho(i\Psi_7, \{c \in C_7 \setminus C_{_{CF,7}} \mid degree_7(c) = 1\})$.

(9) *Increment lower bound and decrement upper bound of configured range connectors:*
$i\Psi_9 = i\Psi_8$ *except*
$l^{N}_{_{C,9}} = l^{N}_{_{C,8}} \oplus$
$(\{(c, (lwb_8(c) + \pi^i(c), upb_8(c) - \pi^d(c))) \mid c \in C_8 \cap (C^{_C} \setminus C^{_C}_{_{CF}}) \wedge upb_8(c) \neq k\}$
$\cup$
$\{(c, (lwb_8(c) + \pi^i(c), degree_8(c) - \pi^d(c))) \mid x \in C_8 \cap (C^{_C} \setminus C^{_C}_{_{CF}}) \wedge lwb_8(c) \neq k \wedge upb_8(c) = k\})$.

(10) *Align lower and upper bound of range connectors with potential change in degree:*
$i\Psi_{10} = i\Psi_9$ *except*
$l^{N}_{_{C,10}} = l^{N}_{_{C,9}} \oplus$
$(\{(c, (degree_9(c), upb_9(c))) \mid c \in C_9 \setminus C_{_{CF}} \wedge lwb_9(c) > degree_9(c) \wedge (upb_9(c) \leq degree_9(c) \vee upb_9(c) = k)\}$
$\cup$
$\{(c, (lwb_9(c), degree_9(c))) \mid c \in C_9 \setminus C_{_{CF}} \wedge lwb_9(c) \leq degree_9(c) \wedge upb_9(c) > degree_9(c) \wedge upb_9(c) \neq k)\}$
$\cup$
$\{(c, (degree_9(c), degree_9(c))) \mid c \in C_9 \setminus C_{_{CF}} \wedge lwb_9(c) > degree_9(c) \wedge upb_9(c) > degree_9(c) \wedge upb_9(c) \neq k)\})$.

(11) *Apply configuration to remaining roles, objects and range connectors:*
Let $\sigma \in \{C, R, O\}$. *Then* $i\Psi_{11} = i\Psi_{10}$ *except*
$l^{M}_{\sigma,11} = l^{M}_{\sigma,10} \oplus$
$\{(x, \pi^{M}(x)) \mid x \in N_{10} \cap (dom(\mathcal{C}_{_R}) \cup dom(\mathcal{C}_{_O}) \cup (dom(\mathcal{C}_{_C}) \setminus dom(\mathcal{C}_{_{C_{CF}}})))\}$,
$l^{U}_{11} = l^{U}_{10} \oplus \{(x, \pi^{U}(x)) \mid x \in N_{10} \cap dom(\mathcal{C}_{_{IN}})\}$, *and*
$nm_{11} = nm_{10} \oplus \{(ro, \pi^{S}(ro)) \mid ro \in N_{10} \cap (dom(\mathcal{C}_{_R}) \cup dom(\mathcal{C}_{_O}))\}$.

(12) *Replace functions without mandatory role assignment with arcs (this may result in consecutive events to be removed):*
Let $F_{_X} = \{f \in F_{_{N,11}} \mid \nexists_{r \in {}^{\bullet}_{\bullet}f^{R}} [l^{M}_{R}(r) = MND] \wedge \nexists_{c \in {}^{\bullet}_{\bullet}f^{R}} [l^{M}_{C}(c) = MND]\}$.
Then $i\Psi_{12} = \delta(\Lambda_{_E}(\varrho(i\Psi_{11}, F_{_X})), \Omega(i\Psi_{11}, F_{_X}))$.

(13) *Replace sese control-flow connectors with arcs (this may result in consecutive functions to be interleaved with events and consecutive events to be removed):*
$i\Psi_{13} = \Lambda_{_E}(\Lambda_{_F}(\Lambda_{_C}(i\Psi_{12})))$.

(14) *Insert connectors to bypass optional functions:*

$$\beta_{i\Gamma}(i\Gamma, \mathcal{C}_{i\Gamma}) = i\Psi_{14} = \varphi(i\Psi_{13}, \{f \in F_{13} \mid \mathcal{C}_F(f) = OPT\}).$$

Figure 11 shows how the individualization algorithm is applied to a fragment of the audio editing process model of Figure 6, according to the configuration described in Section 4.2 (editing a feature movie without music on tape). The first model on the top-left corner shows the result of applying steps 2 and 3 of the algorithm. Since the XOR-split after function Progress update was configured to the SEQ_n starting with event Changes not required, step 2 removes the arc linking the split with this event. In step 3 this event and its outgoing arc to the XOR-join after event Spotting session are removed from the model, as they are no longer on a path from the original start event Shooting finished, to the original end event Editing finished (not shown in the fragment). The second model illustrates the result of applying step 4 of the algorithm, where functions Music design and Process update have been replaced by an arc and their roles and objects have been removed altogether, because these functions were switched *OFF*. This change does not generate any sequence of events so no further operation is performed in this step. Step 5 does not produce any change in the model since no range connectors were switched *OFF*.

The third model is the result of applying step 6. Here roles Composer and Assistant Director and output objects Music cues, Effect cues and Atmos cues, are removed from function Spotting session, while input objects Effect cues and Atmos cues, and output objects Effect tracks and Atmos tracks are removed from function Sound design. This is because the movie featured music only, so the above roles and objects were not required and thus were switched *OFF*. Step 7 does not produce any change in the model since after step 6 all range connectors are still linked to at least one role or object. The next model is the result of applying steps 8 and 9. Step 8 replaces each range connector that remained with a degree of one after step 6, with one arc. Step 9 increments the lower bound of the OR-join $(1 : k)$ connector for the roles of function Sound design from 1 to k, because both roles were required to partake in this function. For convenience, this range is thus shown as an AND. Similarly, the role connector $2 : k$ of function Spotting session is shown as an AND, given that $k : k$ is the only possible range after removing Assistant Director. Step 10 does not change the model since there is no misalignment in the range of the remaining range connectors after steps 6 and 9.

The fifth model is the result of applying step 11, where all roles and objects not switched *OFF* are assigned their configuration values. Role Producer for Spotting session is left as it is since it is not specialized whereas role Editor for Sound design is set to Sound Editor. All occurrences of object Picture cut are set to Tape since the project is shot on tape, and all occurrences of object Dialog cues are set to Paper. Moreover, all configurable roles and objects are left as *MND*. Finally, the last model is the result of applying step 13, where

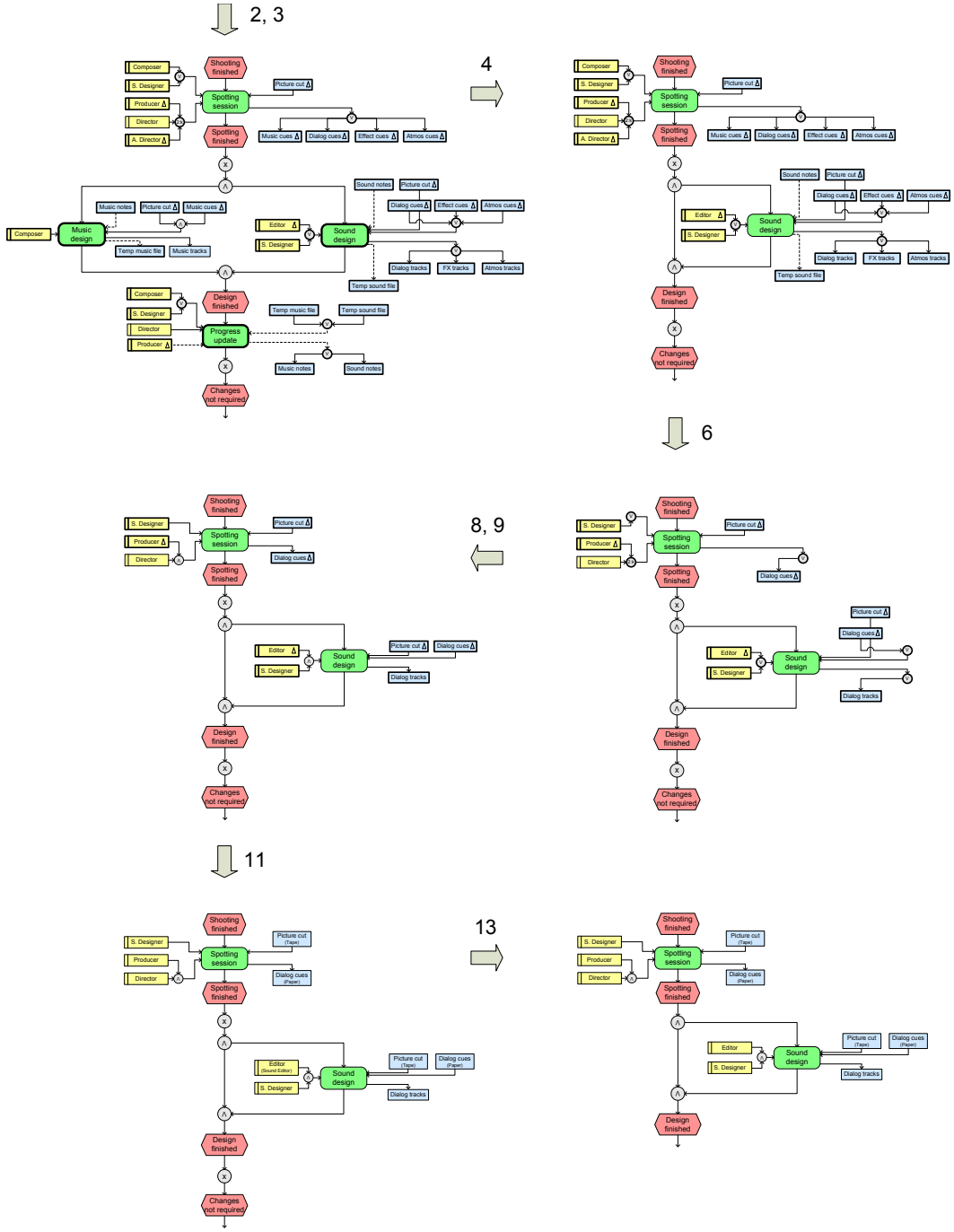

Fig. 11. The application of the individualization algorithm to a fragment of the audio editing process model of Figure 6.

the two sese control-flow XOR connectors are replaced by arcs. As a result, the sequence of events Design finished and Changes not required is produced. This leads to event Changes not required being removed as well. We can observe that this model is a fragment of the individualized model shown in Figure 7.

The definition of syntactical correctness also applies to C-iEPCs. A C-iEPC is syntactically correct if and only if its iEPC component is syntactically correct. The following theorem states that the (C-)iEPC $\beta_{i\Gamma}(i\Gamma, \mathcal{C}_{i\Gamma})$ resulting from the application of the individualization algorithm is syntactically correct if the initial C-iEPC is syntactically correct.

Theorem 13 *Let $i\Gamma$ be a syntactically correct C-iEPC and $\mathcal{C}_{i\Gamma}$ be one of its configurations. Then $\beta_{i\Gamma}(i\Gamma, \mathcal{C}_{i\Gamma})$ is a syntactically correct (C-)iEPC.*

PROOF. *See Appendix.*

6 Questionnaire-Based Approach and Tool Support

The extension of process configuration to other process perspectives beyond the control-flow may induce an overhead in the process modeling lifecycle. For example, as shown in Section 4.2, there can be interdependencies among different process perspectives dictated by the application domain and by the structural requirements of the adopted notation. If, on the one hand, structural requirements are automatically fulfilled by the individualization algorithm, on the other hand domain requirements may prevent users from freely setting configurable nodes.

To cope with this, we propose to configure C-iEPC process models via a questionnaire-based approach, which we presented in previous work [22,25]. The idea is to encode domain requirements with *constraints* over domain choices, in order to abstract from the specific process modeling notation adopted. These choices capture the variability of the application domain (i.e. all the available domain options) and form the answers to a set of questions. For example, a question for audio post-production could be "What audio elements will be used in the project?" and the answers could be "Music", "Dialog", "Effect" and "Atmos". Specifically, each choice is encoded by a *domain fact*, which is a boolean variable that can be set to *true* or *false*, while domain constraints are encoded as boolean conditions over the values of domain facts.

Questions and their domain facts are organized in a questionnaire model. The link between configurable process models and questionnaire models is achieved by mapping each process variant to a condition over the values of domain facts, such that when the condition holds, the specific variant is selected in the process model. In this way process configuration can be achieved by answering a questionnaire expressed in natural language, thus masking the complexity of the underlying process model to subject-matter experts. For example, answering the above question only with "Dialog" would imply the removal of all those tasks, objects and roles that relate to the editing of music, effects

and atmos, thus obtaining a model similar to the one shown in Figure 7. An extract of the questionnaire model for the process model of Figure 5.2 is shown in Figure 12, where each question is associated with a number of domain facts and identified by a unique identifier.

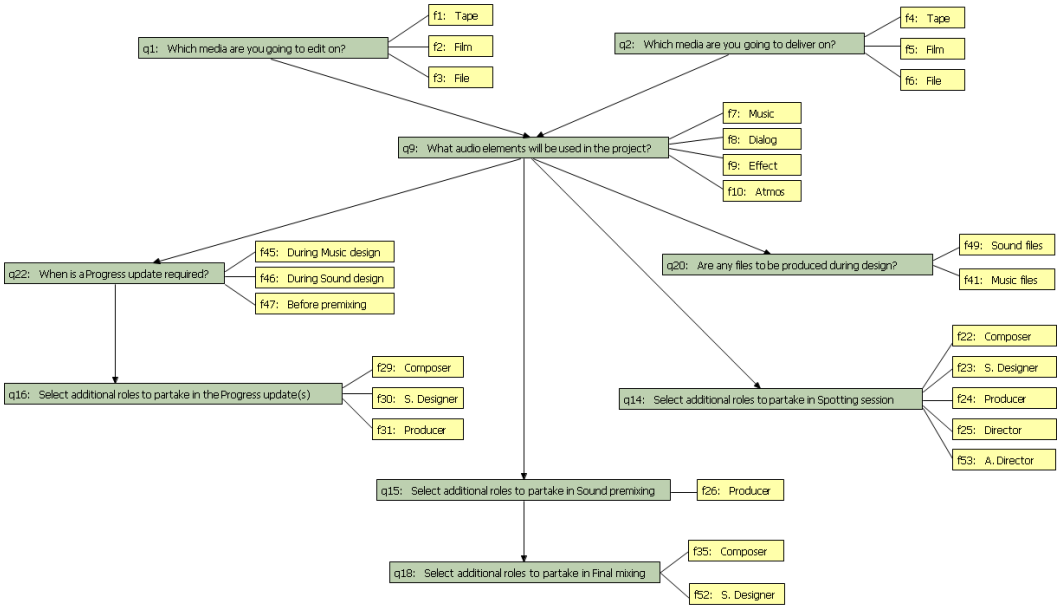

Fig. 12. An extract of the questionnaire model for configuring the audio editing process model.

A questionnaire model also allows one to specify the order dependencies in which questions should be posed to users (indicated by an arrow between questions in Figure 12). In this way the most discriminating questions can be posed first, so as to restrict the answer to subsequent questions on the basis of the domain constraints. For instance, since object Sound files (encoded by fact f_{49} in q_{20}) and role Sound Designer (encoded by f_{23} in q_{14}, f_{30} in q_{16} and f_{52} in q_{18}) are not available if a project does not feature any of Dialog, Effect and Atmos (f_8, f_9 and f_{10} in q_9), we define the domain constraint $\neg(f_8 \vee f_9 \vee f_{10}) \Rightarrow \neg(f_{49} \vee f_{30} \vee f_{23} \vee f_{52})$. Then we set q_9 to be asked before q_{14}, q_{16}, q_{18} and q_{20}, as shown in Figure 12. In this way, if we answer q_9 only with "Music" (f_7), the answer to q_{14}, q_{16}, q_{18} and q_{20} will be automatically restricted (e.g. "Sound Designer" will no longer be available in q_{18}). A question whose answer has been fully determined by answering a preceding question, is no longer relevant and can thus be skipped. This is the case of q_{18} where at least one role between "Composer" (f_{35}) and "Sound Designer" (f_{52}) needs to be chosen to partake in the Final mixing ($f_{35} \vee f_{52}$). After answering q_9 only with "Music", the only possible answer to q_{18} will be $f_{35} = \mathit{true}$ and $f_{52} = \mathit{false}$.

For an in-depth description of the questionnaire-based approach and its link to configurable process models, the reader is referred to [22,25].

The questionnaire-based approach was implemented in an open-source toolset named Synergia [24]. Synergia assists domain experts and process modelers in creating questionnaire models and configurable process models, in mapping questionnaire models to process models, in answering interactive questionnaires and in individualizing process models according to the answers given. In Synergia configurable process models can be defined in C-EPC and C-YAWL [18] (the latter being the configurable extension of the YAWL notation). Figure 13 shows the various tools that compose Synergia and how they interact with each other.

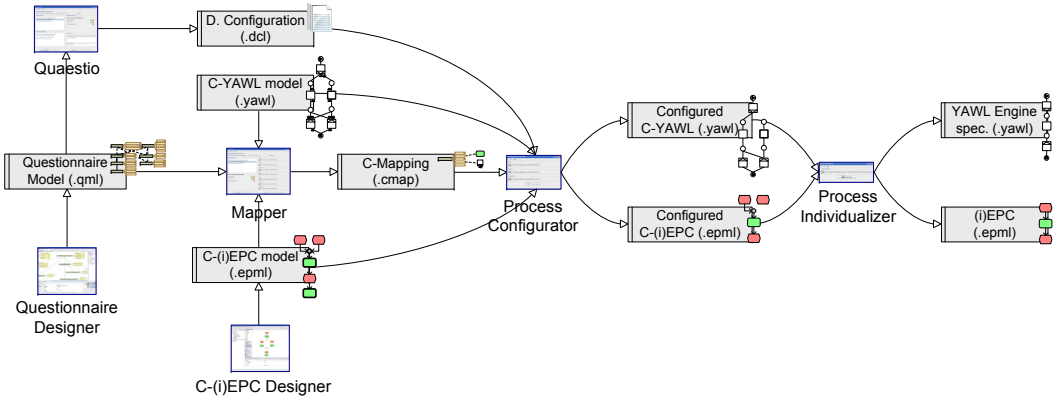

Fig. 13. Synergia – tools interaction map.

The first tool of Synergia is *Questionnaire Designer*, which allows modelers to visually create and validate questionnaire models. For example, this tool can identify circular dependencies among questions that would lead to deadlocks when answering the questionnaire. The generated questionnaire models can then be imported into *Quaestio* to be answered. Quaestio is an interactive questionnaire tool. It poses questions to users in an order consistent with the order dependencies established in the questionnaire model, and prevents users from entering incorrect answers that would violate the domain constraints. The dynamic checking of the domain constraints is achieved by using a SAT solver that relies on an internal representation of the constraints as Shared Binary Decision Diagrams.

In order to use the answers given to a questionnaire (i.e. a domain configuration) to configure a process model, users first need to define a mapping between the questionnaire model and the variation points in the process model. This can be done in the *Mapper* tool. A mapping can then be imported in the *Process Configurator* tool, along with the respective configurable process model and domain configuration, and used to configure the process model.

The last step of the questionnaire-based approach is the individualization of a configured process model. This is done via the *Process Individualizer* tool, which produces a syntactically correct EPC or YAWL model, depending on the language used to describe the configurable process model.

In the course of this research, we have extended the Synergia toolset to cater for the questionnaire-based configuration of C-iEPC process models. Specifically, we have extended the XML serialization format for EPCs (EPML 1.2 [27]) to provide a serialization for roles, objects, range connectors and their variants, as well as for hierarchy models. The new version, namely EPML 2.0 [28], also provides a more concise serialization of some existing EPC elements.

Moreover, we have developed a new tool, namely *C-iEPC Designer*, to allows modelers to visually create C-iEPC models in EPML 2.0. This tool has been realized by extending an existing editor for EPC models (EPC Tools [10]). A screenshot of C-iEPC Designer showing the configurable model for audio post-production is depicted in Figure 14. Among other features, this tool offers a Properties view to specify the attributes of a node, such as the range of a connector or the optionality of a role. Furthermore, the Properties view can be used to configure a C-iEPC model for debugging purposes. This can be done by assigning a value to each configurable dimension of a configurable node. For example, Figure 14 shows the Properties view for the configurable range split associated with the output objects of function Spotting session, where the range has been restricted to the single node Music cues. The tool enforces the configuration requirements specified in Definition 10. So, for example, it is not possible to restrict a mandatory role to optional or to restrict a range connector to a node which is not in its preset or postset. Accordingly, the range split in Figure 14 can only be restricted to one of Music cues, Dialog cues, Effect cues and Atmos cues, as shown by the drop-down menu in the Properties view (property 'Goto').

The Mapper has been extended to import C-iEPC models defined in EPML 2.0. Upon import, this tool identifies the variation points of the C-iEPC model, so that these variation points can be mapped to domain choices from a questionnaire model. The Process Configurator has also been extended to be able to configure a C-iEPC model (in EPML 2.0) according to the answers of a questionnaire while the $\beta_{i\Gamma}$ algorithm has been implemented in the Process Individualizer, to commit the required model transformations and generate a syntactically correct iEPC.

Synergia and the files for the working example can be downloaded from the Process Configuration web-site. [5]

[5] `www.processconfiguration.com`

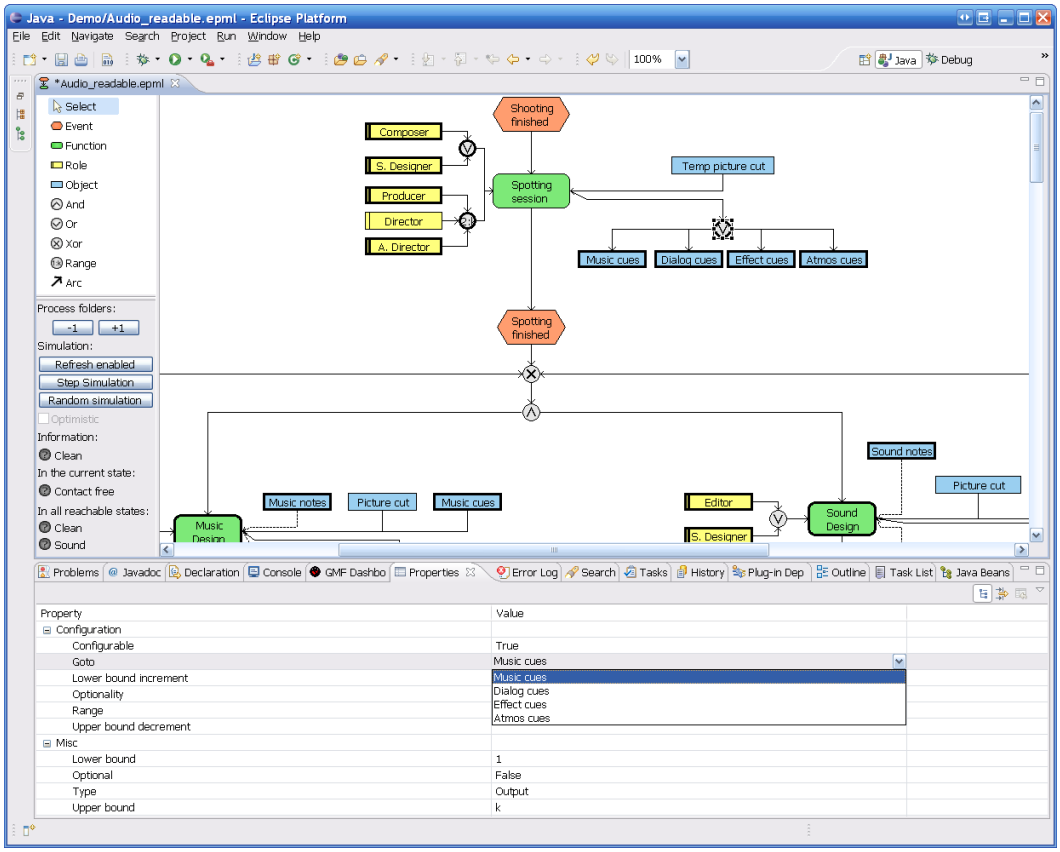

Fig. 14. C-iEPC Designer – showing the model in Figure 6.

7 Post-Production Case Study

The working example used in this paper is a simplified version of a reference process model for post-production that we created in collaboration with members of the AFTRS, over a period of nine months. The AFTRS is an Australian training and research facility for Graduate Diplomas, Master courses and short courses in Film and TV production. This school has engaged, with other stakeholders, in an initiative to capture business process models in the film industry. However, it was quickly noted that process models in this industry have a high degree of variability. Basically, each production project works differently from the others.

In this case study, we focused specifically on process models for the post-production phase. Post-production starts after the shooting phase and deals with the design and edit of the picture, music and sound of a screen project. Creativity is a distinguishing feature of this domain: a single decision made by the director, such as that of not having any music, can radically change the whole post-production. This necessarily leads to a great deal of variability. For this reason, the domain in question was deemed suitable to evaluate our framework.

A number of configurable process models were defined to describe the overall post-production phase, which comprises picture and audio post-production, and its variations. In this phase, we received input from a producer and two sound editors. Given the involvement of the sound editors, we were also able to capture detailed information regarding roles and business objects for the audio post-production process. For this reason we used the C-iEPC notation to model this process, while we used the C-EPC notation to model the process for picture post-production. The complete reference process model consists of 792 process elements (spread across different diagrams), of which 183 are variation points (23% of the total), each allowing a number of process variants for a total of around 310,000 valid individualizations.

For the construction of this model, we first established the control-flow. Then, we identified the objects used and produced by each task (examples are the film roll, the dialogue tracks). Thirdly, we assigned to each task the human roles that have to perform them (the director, the sound designer, etc.). Afterwards, we identified variation points in the process model, in terms of which tasks, objects and roles can vary.

Having defined the process model for post-production, we identified a set of domain facts to capture the choices that need to be taken to configure the reference model, and we grouped them into suitable questions. Firstly, we defined one fact for each factor that yields a high number of process variations. Such factors, like the budget level, correspond to domain decisions which are usually not captured in the process model. Secondly, we encoded each fine-grained decision with one fact. Such decisions have little or no impact on the rest of the system. For example, the type of editing suite only affects the medium format, while both the type of suite and the format are determined by the available budget. Thirdly, we defined a system of constraints to encode the interplay among these facts, and we used the Questionnaire Designer to check the satisfiability. With the help of the tool, we also realized that some fine-grained facts were redundant, as the variants they captured could be determined by the configuration of other facts. This is the case with the Telecine suite, which is a piece of equipment that is only required in particular situations, such as when the shooting is on film and the release is on tape. Thus, it was sufficient to encode all the shooting and release formats, to indirectly capture the Telecine options.

The complete questionnaire model consists of three sub-questionnaires (picture editing, sound editing and screen composition), and an introductory questionnaire linking them together, for a total of 53 questions and 162 facts.

The questionnaire was used to configure the reference process model to the requirements of student projects at the AFTRS. This showed that depending on the context, a customized process model can be generated by domain ex-

perts in a straightforward manner. The generated models can then be used by the members of the school to guide the planning and the actual execution of a screen project. Also, they can be used in the learning environment to teach students who aspire to become editors, sound designers and screen composers about the stages of the post-production process. Furthermore, they can be useful for production and direction students to clearly understand the relations among the various drivers behind post-production, such as budget and schedule. For these reasons, the models produced during this case study and the Synergia toolset are planned to be introduced in the AFTRS syllabus of production and editing courses.

In conclusion, this experience demonstrated that the C-iEPC notation is able to capture complex variability requirements in a process involving roles, business objects and their associations with tasks.

8 Summary and Discussion

This paper addressed a major shortcoming of existing configurable process modeling notations, namely their lack of support for the resource and object perspectives. The main contribution is a notation for configurable process modeling, namely C-iEPCs, that extends the EPC notation with mechanisms for representing a range of variations along multiple perspectives. While embodied in the EPC notation, the proposed extensions are defined in an abstract manner, so that they can be transposed to other notations, such as BPMN and UML ADs.

The proposal was applied to capture a comprehensive reference process model for audio post-production as part of a case study conducted with domain experts from the AFTRS. The validation was supported by a toolset, namely Synergia, that allows designers to capture and to individualize configurable process models using the algorithm presented in this paper.

The research highlighted the intricacies that configurable process modeling across multiple perspectives brings (e.g. removal of a role may entail the removal of a function). To overcome this issue, we proposed to configure process models via a questionnaire-driven approach that we presented in previous work. Specifically, we captured dependencies between variation points via domain constraints encoded in a questionnaire model. Analysts configure the process model by answering this questionnaire. In this way, we guaranteed that the individualized models are domain-compliant. We also provided an individualization algorithm for C-iEPC process models and proved that the syntactic correctness of the individualized iEPCs is guaranteed by this algorithm.

On the other hand, the proposed approach does not prevent users from creating inconsistencies between object-flow dependencies and control-flow dependencies that may lead to behavioral issues, e.g. an object-flow dependency that contradicts a control-flow dependency. As a result, the behavioral correctness of an iEPC is not guaranteed during configuration. In [26], we defined a verification approach for process models including object-flow dependencies, which is based on reachability graph analysis. This approach works in the general case but is exponential in computational cost. To avoid the need for verifying each possible individualization beforehand, in [1] we devised a technique for individualizing configurable process models in a correctness-preserving way. Specifically, this technique allows us to automatically infer a set of *process constraints* from the control-flow of a configurable process model that, if satisfied during configuration, guarantee the syntactic correctness of the individualized model. We proved that for *free-choice* nets [14] (a large class of process models), these constraints also ensure that behavioral correctness is preserved. In future work we plan to extend these results to cover object-flow dependencies, by investigating which object-flow dependencies can be mapped to free choice constructs.

Another direction for future work is to investigate techniques for automating the construction of configurable process models. A possible starting point is to collect a number of related process models from different process design projects, and to merge them together. Since process models are usually represented as graphs, algorithms from the field of graph matching could prove beneficial [9]. These algorithms could be employed to identify a common denominator among all models, and variants with respect to such a common denominator. These variants could then be captured as configurable nodes on top of this common denominator. Another option to automate the construction of configurable process models would be to use process mining techniques. The idea of process mining is to take event logs related to a business process and to derive a process model that matches the event log in question. In [20] the authors discuss extensions to existing process mining techniques that allow one to derive a C-EPC from a regular EPC and one or several logs. Further research is required to extend these techniques beyond the control-flow perspective and to validate their applicability in practice.

Acknowledgments. We thank Florian Gottschalk for his input to a previous version of this paper and the AFTRS team for their contribution to the design of the reference model. This research is funded by the ARC Discovery Project "Next Generation Reference Process Models" and the Estonian Centre of Excellence in Computer Science.

References

[1] W.M.P. van der Aalst, M. Dumas, F. Gottschalk, A.H.M. ter Hofstede, M. La Rosa, and J. Mendling. Preserving Correctness During Business Process Model Configuration. *Formal Aspects of Computing*, 2009.

[2] W.M.P. van der Aalst and K. M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[3] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.

[4] D.S. Batory and B.J. Geraci. Composition Validation and Subjectivity in GenVoca Generators. *IEEE Transactions on Software Engineering*, 23(2):67–84, 1997.

[5] J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, and D. Kuropka. Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In M. Khosrow-Pour, editor, *Proceedings of the 14th Information Resources Management Association International Conference*, pages 615–619. IRM Press, 2004.

[6] J. Becker, P. Delfmann, and R. Knackstedt. Adaptive Reference Modeling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In J. Becker and P. Delfmann, editors, *Reference Modeling*, pages 27–58. Springer, 2007.

[7] B. Benneth. Kiss or Kill. Feature movie. Australia, 1997.

[8] 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*, 2(1):65–104, 1999.

[9] H. Bunke. Recent Developments in Graph Matching. In A. Sanfeliu, J.J. Villanueva, M. Vanrell, R. Alquezar, A.K. Jain, and J. Kittler, editors, *Proceedings of the 15th International Conference on Pattern Recognition (ICPR'00)*, volume 2, pages 117–124. IEEE Computer Society, 2000.

[10] N. Cuntz and E. Kindler. EPC Tools. Home Page. `http://wwwcs.uni-paderborn.de/cs/kindler/research/EPCTools`. Accessed: September 2009.

[11] T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.

[12] K. Czarnecki and M. Antkiewicz. Mapping Features to Models: A Template Approach Based on Superimposed Variants. In R. Glück and M.R. Lowry, editors, *Proceedings of the 4th International Conference on Generative Programming and Component Engineering*, volume 3676, pages 422–437. Springer, 2005.

[13] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[14] J. Desel and J. Esparza. *Free Choice Petri Nets*. Vol. 40 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1995.

[15] G. Engels, A. Förster, R. Heckel, and S. Thöne. Process Modeling Using UML. In M. Dumas, A.H.M. ter Hofstede, and W.M.P. van der Aalst, editors, *Process Aware Information Systems: Bridging People and Software Through Process Technology*, pages 85–118. Wiley, 2005.

[16] 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.

[17] F. Gottschalk, W.M.P. van der Aalst, and M.H. Jansen-Vullers. Configurable Process Models – A Foundational Approach. In J. Becker and P. Delfmann, editors, *Reference Modeling*, pages 59–78. Springer, 2007.

[18] 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.

[19] S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, 1996.

[20] M.H. Jansen-Vullers, W.M.P. van der Aalst, and M. Rosemann. Mining Configurable Enterprise Information Systems. *Data & Knowledge Engineering*, 56(3):195–244, 2006.

[21] E. Kavakli and P. Loucopoulos. Experiences with goal-oriented modelling of organisational change. *IEEE Transactions on Systems, Man and Cybernetics – Part C*, 36(2):221–235, 2006.

[22] M. La Rosa, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Questionnaire-based Variability Modeling for System Configuration. *Software and Systems Modeling*, 8(2), 2009.

[23] M. La Rosa, M. Dumas, A.H.M. ter Hofstede, J. Mendling, and F. Gottschalk. Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects. In Q. Li, S. Spaccapietra, E. Yu, and A. Olivé, editors, *Proceedings of the 27th International Conference on Conceptual Modeling (ER'08)*, volume 5231 of *Lecture Notes in Computer Science*, pages 199–215. Springer, 2008.

[24] M. La Rosa, F. Gottschalk. Synergia – Comprehensive Tool Support for Configurable Process Models. In A.K. Alves de Medeiros and B. Weber, editors, *Proceedings of the BPM'09 Demonstration Track*, volume 489 of CEUR-WS, CEUR, 2009. `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-489`

[25] M. La Rosa, F. Gottschalk, M. Dumas, and W.M.P. van der Aalst. Linking Domain Models and Process Models for Reference Model Configuration. In A.H.M. ter Hofstede, B. Benatallah, and H.-Y. Paik, editors, *BPM'07 Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 417–430. Springer, 2008.

[26] J. Mendling, M. La Rosa, and A.H.M. ter Hofstede. Correctness of Business Process Models with Roles and Objects. QUT ePrints 13172. Queensland University of Technology, Brisbane, Australia. 2008.

[27] J. Mendling and M. Nüttgens. EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). *Information Systems and e-Business Management*, 4(3):245–263, 2006.

[28] J. Mendling, M. Nüttgens, and M. La Rosa. EPML Schema 2.0. `http://www.processconfiguration.com/schemas/EPML_2.0.xsd`. Accessed: September 2009.

[29] C. Menzel and R.J. Mayer. The IDEF Family of Languages. *Handbook on Architectures of Information Systems*, Part one:215–249, 1998.

[30] OASIS. *Web Services Business Process Execution Language (WS-BPEL), Version 2.0. OASIS Standard.* OASIS, 2007. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf`. Accessed: September 2009.

[31] Object Management Group. *Business Process Modeling Notation (BPMN), Version 1.2.* OMG, 2009. `http://www.omg.org/spec/BPMN/1.2`. Accessed: September 2009.

[32] K. Pohl, G. Böckle, and F. van der Linden. *Software Product-line Engineering – Foundations, Principles and Techniques.* Springer, 2005.

[33] M. Razavian and R. Khosravi. Modeling Variability in Business Process Models Using UML. In S. Latifi, editor, *Proceedings of the 5th International Conference on Information Technology: New Generations (ITGN'08)*, pages 82–87. IEEE Computer Society, 2008.

[34] M. Reichert and P. Dadam. ADEPT$_{flex}$: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[35] H.A. Reijers, R.S. Mans, and R.A. van der Toorn. Improved Model Management with Aggregated Business Process Models. *Data & Knowledge Engineering*, 68(2):221–243, 2009.

[36] I. Reinhartz-Berger, P. Soffer, and A. Sturm. A Domain Engineering Approach to Specifying and Applying Reference Models. In J. Desel and U. Frank, editors, *Workshop Enterprise Modelling and Information Systems Architectures*, volume 75 of *Lecture Notes in Informatics*, pages 50–63. GI, 2005.

[37] M. Rosemann and W. M. P van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, 2007.

[38] 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. Falcão 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, 2005.

[39] A.-W. Scheer. *ARIS – Business Process Frameworks*. Springer, 3rd edition, 1999.

[40] A. Schnieders and F. Puhlmann. Variability Mechanisms in E-Business Process Families. In W. Abramowicz and H.C. Mayr, editors, *Proceedings of the 9th International Conference on Business Information Systems (BIS'06)*, volume 85 of *Lecture Notes in Informatics*, pages 583–601. GI, 2006.

[41] S. Stephens. The Supply Chain Council and the Supply Chain Operations Reference Model. *Supply Chain Management – An International Journal*, 1(1):9–13, 2001.

Appendix

In order to prove Theorem 13 from Section 5.2, we first define the following abbreviations:

- *ee* iEPC, as an iEPC which may have consecutive events (thus relaxing the Arc relation of Definition 3),
- *ff* iEPC, as an iEPC which may have consecutive functions (thus relaxing the Arc relation of Definition 3),
- *dn* iEPC, as an iEPC which may have control-flow nodes not on some path from a start event to an end event (thus relaxing requirement 1 of Definition 5),
- $ss(c)$ iEPC, as an iEPC which may have sese control-flow connectors (thus relaxing requirement 5a of Definition 5),
- $ss(r)$ iEPC, as an iEPC which may have sese role connectors (thus relaxing requirement 5b of Definition 5),
- $ss(i)$ iEPC, as an iEPC which may have sese input connectors (thus relaxing requirement 5c of Definition 5),
- $ss(o)$ iEPC, as an iEPC which may have sese output connectors (thus relaxing requirement 5d of Definition 5),
- *dc* iEPC, as an iEPC which may have coronas that are not connected to functions (thus relaxing requirements 5b, 5c, 5d, 6 and 7 of Definition 5).
- *nmr* iEPC, as an iEPC which may have functions without mandatory role assignment (thus relaxing requirement 8 of Definition 5).
- *mrb* iEPC, as an iEPC which may have misaligned range connector bounds (thus relaxing requirement 10 of Definition 5).

We now list some properties of the operators of Definition 11 which are used in the proof of Theorem 13. Given an operator *op* and one of the above abbreviations *abbr*, with $\overline{abbr}$ we indicate that the property of the iEPC *abbr* refers to, remains unchanged after applying *op*. For example, given an *ff* iEPC, after applying *op*, with $\overline{ff}$ we indicate that the iEPC produced by *op* does not generate new consecutive functions.

Proposition 14 (Operator Properties) *Let $i\Upsilon$ be a syntactically correct ee-ff-ss(c) iEPC and let $X \subset N_{CF}$ be a set of control-flow nodes. The following properties hold for the operators in Definition 11:*

(a) *if $X \subset F$, then $\varrho(i\Upsilon, X)$ is a syntactically correct ee-$\overline{ff}$-ss(c) iEPC.*[6]
(b) *if $X = \{n \in N_{CF} \mid \nexists_{\phi \in N_{CF}^+, e_s \in E_s, e_e \in E_e, \phi: e_s \hookrightarrow e_e} [n \in \alpha(\phi)]\}$, i.e. if X is the set of all control-flow nodes not on a path from a start event to an end event, then:*
  (i) *$\varrho(i\Upsilon, X)$ is a syntactically correct $\overline{ee}$-$\overline{ff}$-ss(c)-dc iEPC.*
  (ii) *$\varrho(i\Upsilon, X) = \delta(i\Upsilon, X)$.*
(c) *if $X \subset F$, then $\varphi(i\Upsilon, X)$ is a syntactically correct $\overline{ee}$-$\overline{ff}$-ss($\overline{c}$) iEPC.*
(d) *$\Lambda_E(i\Upsilon)$ is a syntactically correct $\overline{ff}$-ss($\overline{c}$) iEPC.*
(e) *$\Lambda_C(i\Upsilon)$ is a syntactically correct ee-ff iEPC.*
(f) *$\Lambda_F(i\Upsilon)$ is a syntactically correct $\overline{ee}$-ss($\overline{c}$) iEPC.*
(g) *let $i\Upsilon' = \delta(\Lambda_E(\varrho(i\Upsilon, X)), \Omega(i\Upsilon, X))$:*
  (i) *if $X \subset F$, then $i\Upsilon'$ is a syntactically correct $\overline{ff}$-ss(c) iEPC.*
  (ii) *if $i\Upsilon$ is also nmr and $X = \{f \in F \mid \nexists_{r \in {}^{\bullet}f^R} [l_R^M(r) = MND] \wedge \nexists_{c \in {}^{\bullet}f^R} [l_C^M(c) = MND]\}$, i.e. if X is the set of all functions without mandatory role assignment, then $i\Upsilon'$ is a syntactically correct $\overline{ff}$-ss(c) iEPC.*
  (iii) *if $i\Upsilon$ is also dn and X is the set of all control-flow nodes not on some path from a start event to an end event, then $i\Upsilon'$ is a syntactically correct $\overline{ff}$-ss(c) iEPC.*

PROOF.

(a) *New consecutive functions can only be generated by removing an event or a control-flow connector between two functions. However X contains functions only. Proving that the requirements of Definition 5 are not violated, apart from creating new sese control-flow connectors, is straightforward.*
(b.i) *New consecutive functions can only be generated by removing an event or control-flow connector that is between two functions. We show that such a node cannot be removed. Let $f_1, f_2 \in F$ be two functions such that f_1 is on a path from a start event e_s, f_2 is on a path to an end event e_e, and*

[6] New sese control-flow connectors may be generated after merging sibling branches, i.e. branches sharing the starting split and the completing join, if these branches consist of removed functions only.

there exists a node $n \in N_{CF} \setminus F$ *such that* $f_1 \in {}^{CF}\bullet\, n$ *and* $f_2 \in n\, {}^{CF}\bullet$. *Assume* $n \in X$. *Then* n *is on a path from* e_s *to* e_e *(Contradiction). Hence,* $n \notin X$. *The case of consecutive events is similar. Proving that the requirements of Definition 5 are not violated, apart from creating new sese control-flow connectors and disconnected coronas, is straightforward.*

(b.ii) *Let* $A^X = \{(a,b) \in N_{CF} \setminus X \times N_{CF} \setminus X \mid \exists_{x,y \in X, \phi \in X^+, \phi: x \overset{*}{\hookrightarrow} y} [a \in {}^{CF}\bullet\, x \wedge b \in y\, {}^{CF}\bullet]\}$ *be the set of arcs added by* ϱ. *Assume* $A^X \neq \varnothing$. *Let* $(\alpha, \beta) \in A^X$, *then for some* $x, y \in X$ *with* $x \overset{*}{\hookrightarrow} y$, $\alpha \in {}^{CF}\bullet\, x$ *and* $\beta \in y\, {}^{CF}\bullet$. *As* $\alpha, \beta \notin X$, α *is on a path from a start event* e_s *and* β *is on a path to an end event* e_e. *So* x *and* y *are on a path from* e_s *to* e_e *(Contradiction). Hence,* $A^X = \varnothing$.

(c) $\varphi(i\Upsilon, X)$ *does not remove nodes and does not add events or functions. Hence, new consecutive events and new consecutive functions cannot be generated. Also,* $\varphi(i\Upsilon, X)$ *does not remove arcs. Hence, new sese control-flow connectors cannot be generated. Proving that the remaining requirements of Definition 5 are not violated is straightforward.*

(d-f) *Straightforward.*

(g.i) $\varrho(i\Upsilon, X)$ *is a syntactically correct ee-*$\overline{ff}$*-ss*(c) *iEPC (Proposition 14a). Hence,* $\Lambda_E(\varrho(i\Upsilon, X))$ *is a syntactically correct* $\overline{ff}$*-ss*(c) *iEPC (Proposition 14d). Removing the corona of functions that are also removed does not affect syntactic correctness. Thus,* $\delta(\Lambda_E(\varrho(i\Upsilon, X)), \Omega(i\Upsilon, X))$ *is a syntactically correct* $\overline{ff}$*-ss*(c) *iEPC.*

(g.ii) *Similar to* g.i *but all functions without mandatory role assignment are removed.*

(g.iii) $\varrho(i\Upsilon, X)$ *is a syntactically correct* $\overline{ee}$*-*$\overline{ff}$*-ss*(c)*-dc iEPC (Proposition 14b.i). Hence,* $\Lambda_E(\varrho(i\Upsilon, X))$ *is a syntactically correct* $\overline{ff}$*-ss*(c)*-dc iEPC (Proposition 14d). Removing the corona of functions that are also removed does not affect syntactic correctness. Thus,* $\delta(\Lambda_E(\varrho(i\Upsilon, X)), \Omega(i\Upsilon, X))$ *is a syntactically correct* $\overline{ff}$*-ss*(c) *iEPC.*

We recall Theorem 13 and prove it.

Theorem 13 *Let* $i\Gamma$ *be a syntactically correct C-iEPC and* $\mathcal{C}_{i\Gamma}$ *be one of its configurations. Then* $\beta_{i\Gamma}(i\Gamma, \mathcal{C}_{i\Gamma})$ *is a syntactically correct (C-)iEPC.*

PROOF. *We show that the properties of Definition 5 hold for* $\beta_{i\Gamma}(i\Gamma, \mathcal{C}_{i\Gamma})$ *by discussing the different steps of the algorithm.*

(1) $i\Psi_1$ *is a syntactically correct (C-)iEPC.*

(2) Changing connector labels does not affect syntactic correctness. Removing non-SEQ_n *arcs may result in nodes that are no longer on a path from an original start event to an original end event, and in sese control-flow connectors. Hence,* $i\Psi_2$ *is a syntactically correct dn-ss*(c) *(C-)iEPC.*

(3) Observe $i\Psi_3 = \delta(\delta(i\Psi_2, N_X), \Omega(i\Psi_2, N_X)) = \delta(\varrho(i\Psi_2, N_X), \Omega(i\Psi_2, N_X))$ *(Proposition 14b.ii)* $= \delta(\Lambda_E(\varrho(i\Psi_2, N_X)), \Omega(i\Psi_2, N_X))$ *as* $\varrho(i\Psi_2, N_X)$

does not have consecutive events. Hence, as $i\Psi_2$ does not have consecutive functions, $i\Psi_3$ is a syntactically correct $ss(c)$ (C-)iEPC (Proposition 14g.iii).

(4) $i\Psi_4$ is still a syntactically correct $ss(c)$ (C-)iEPC (Proposition 14g.i).

(5) This step may create functions without any mandatory role assignment. Thus $i\Psi_5$ is a syntactically correct $ss(\overline{c})$-nmr (C-)iEPC.

(6) This step may create syntactic issues with range connector cardinality, mandatory role assignment of functions, and range connector bounds. Hence, $i\Psi_6$ is a syntactically correct $ss(\overline{c}, r, i, o)$-nmr-mrb (C-)iEPC.

(7-8) Step 7 resolves those cases related to $ss(r, i, o)$ where the degree of the range connector is zero, while step 8 resolves those cases where the degree is one. Hence, $i\Psi_8$ is a syntactically correct $ss(\overline{c})$-nmr-mrb (C-)iEPC.

(9) Misalignment issues with upper bounds are resolved if, after the decrement, the new upper bound is equal to or below the degree of a range connector. However, misalignments of lower bounds are not resolved. Hence, $i\Psi_9$ is still a syntactically correct $ss(\overline{c})$-$\overline{nmr}$-mrb (C-)iEPC.

(10) All range bound misalignments are resolved. Hence, $i\Psi_{10}$ is a syntactically correct $ss(\overline{c})$-$\overline{nmr}$ (C-)iEPC.

(11) Changing role, object and range connector labels does not lead to new syntactic violations. However, some functions that did not have a mandatory role assignment before, may now have at least one. Hence, $i\Psi_{10}$ is still a syntactically correct $ss(\overline{c})$-nmr (C-)iEPC.

(12) $i\Psi_{12}$ is a syntactically correct $ss(\overline{c})$ (C-)iEPC (Proposition 14g.ii).

(13) $\Lambda_{C}(i\Psi_{12})$ is a syntactically correct ee-ff (C-)iEPC (Proposition 14e). Hence, $\Lambda_{F}(\Lambda_{C}(i\Psi_{12}))$ is a syntactically correct $\overline{ee}$ (C-)iEPC (Proposition 14f). Thus, $\Lambda_{E}(\Lambda_{F}(\Lambda_{C}(i\Psi_{12})))$ is a syntactically correct (C-)iEPC (Proposition 14d).

(14) $i\Psi_{14}$ is a syntactically correct (C-)iEPC (Proposition 14c).