



## COVER SHEET

---

**This is the author version of article published as:**

*Mending, Jan and Recker, Jan C. and Rosemann, Michael and van der Aalst, Wil M. (2005) Towards the Interchange of Configurable EPCs: An XML-based Approach for Reference Model Configuration. In Desel, Jorg and Frank, Ulrich, Eds. Proceedings Enterprise Modelling and Information Systems Architectures, Proceedings of the Workshop in Klagenfurt 75, pages pp. 8-21, Klagenfurt, Austria.*

**Copyright 2005 (please consult author)**

**Accessed from <http://eprints.qut.edu.au>**

# Towards the Interchange of Configurable EPCs: An XML-based Approach for Reference Model Configuration

Jan Mending<sup>1</sup>, Jan Recker<sup>2</sup>, Michael Rosemann<sup>2</sup>, Wil van der Aalst<sup>3</sup>

<sup>1</sup>Vienna University of Economics and Business Administration

jan.mending@wu-wien.ac.at

<sup>2</sup>Queensland University of Technology

{j.recker,m.rosemann}@qut.edu.au

<sup>3</sup>Eindhoven University of Technology

w.m.p.v.d.aalst@tm.tue.nl

**Abstract:** Recent research has led to proposals for the model-driven configuration of Enterprise Systems (ES) by the help of configurable reference models. While the proposed *Configurable EPCs* (C-EPCs) provide adequate conceptual support towards reference model configuration, the issue of translating the configured models towards executable process specifications has not been approached yet. A first step in this direction is the definition of a machine-readable format for C-EPCs that can be used as an interchange format and as an input format for transformations. This paper proposes a C-EPC representation in XML based on the *EPC Markup Language* (EPML) format. We take the formal C-EPC syntax definition as a starting point to define the requirements for a respective extension to EPML and introduce a C-EPC representation in EPML. Furthermore, we introduce the *C-EPC Validator*, a program that generates validity reports for C-EPCs represented in EPML. The C-EPC Validator can be used to validate both configurable and configured C-EPCs. Finally, we highlight future application areas of the C-EPC schema.

## 1 Introduction

Enterprise Systems (ES) support and enhance organizations in their business operations if, and only if, they are well-configured as to the specific organizational requirements. This configuration process is not only time- and resource-consuming but moreover proven to result in severe business failure if conducted badly [Da98]. This notion of misalignment is predominantly visible in the process dimension, i.e. the (mis-) alignment of IT functionality to the actual business processes of an organization [LLO93]. In many cases, it is observed that the system hampers the normal way of handling processes instead of supporting it. This is surprising given the fact that business process orientation as a concept has been a major topic in both academia and practice at least since the 1990's [HC93].

The popularity of business process management is steadily rising, in general terms, leading to a proliferation of various process modeling approaches. This heterogeneity of proprietary process modeling schemas and formats has been the major motivation for the

definition of EPML (*EPC Markup Language*) [MN04, MN05] in order to facilitate the communication and interchange between various business process modeling tools. In this paper, we will foremost consider *Event-Driven Process Chains* (EPC) [KNS92] due to their wide-spread use for reference modeling in the context of Enterprise Systems (cf. e.g. SAP and its reference model that is defined using EPCs [CKL97]).

The fact that most Enterprise Systems are extensively depicted in their reference models motivates the idea of utilizing such reference models for the task of systems configuration to allow for a more intuitive, model-driven approach. We have been developing a reference modeling approach that considers the configurable nature of an Enterprise System. The representation language of this approach is called a *Configurable EPC* (C-EPC). While previous research focused on the theoretical development of the meta model and the notation of C-EPCs [RvdA05], this paper explores the technical stage of ES configuration by outlining an approach how to represent C-EPC configuration in a machine-readable XML syntax. Such a schema may serve as a starting point for transformations towards executable process specifications, forming the basis for systems workflow implementations.

To be more concise, the *aim of our paper* is the specification of C-EPCs in an XML schema that builds on the established EPML format. Such EPML representation of C-EPCs is required in order to support various scenarios of reference model configuration including (a) the XML schema-based evaluation of the validity of configurations, (b) the implementation of an EPML tool for the automatic translation of C-EPCs to regular EPCs, (c) the transformation of C-EPCs to other process specifications, and (d) the facilitation of modeling C-EPCs in existing tools such as ARIS Toolset or EPC Tools.

Addressing this research objective, the remainder of our paper is structured as follows. In Section 2 we present the conceptual background of our research by briefly outlining the basic notions of C-EPCs and EPML. Section 3 discusses the formal specification of the C-EPC syntax as an extension to the EPML schema. We then report on the evaluation of C-EPCs as to the validation of lawful configurations. In this context, we introduce a validation program called *C-EPC Validator*. After discussing related research efforts in Section 4 we conclude the paper in Section 5 and give an outlook on future research.

## 2 Configurable EPCs and EPML

### 2.1 On the Notion of C-EPCs

Application reference models are frequently used by Enterprise System vendors to highlight and explain the functionality of their software packages. However, such reference models are rarely deployed in the process of configuring Enterprise Systems, which is surprising given that the main objective of reference models is to streamline the design of particular models. Yet, the transformation (better: configuration) of reference models to individual models is so far at best scarcely supported through the underlying reference model languages. The lack of language expressiveness denotes a major issue for reference model users, as reference models fail to depict configuration alternatives and do not

provide adequate decision support towards the selection of configuration alternatives.

Addressing these issues, this section introduces *Configurable EPCs* (C-EPCs) as a representation form for configurable reference models. In order to embody configuration-related information in conceptual process models, the traditional EPC [KNS92] had to be extended, leading to the definition of the C-EPC. Its notion has been introduced and formalized in [RvdA05], therefore we only discuss the basic notation here.

C-EPCs may be used to depict two statuses of a reference model: (a) as a model depicting configuration *alternatives* (such as the one depicted in the left part of Figure 1), and (b) as a model depicting configuration *selections* (right part of Figure 1).

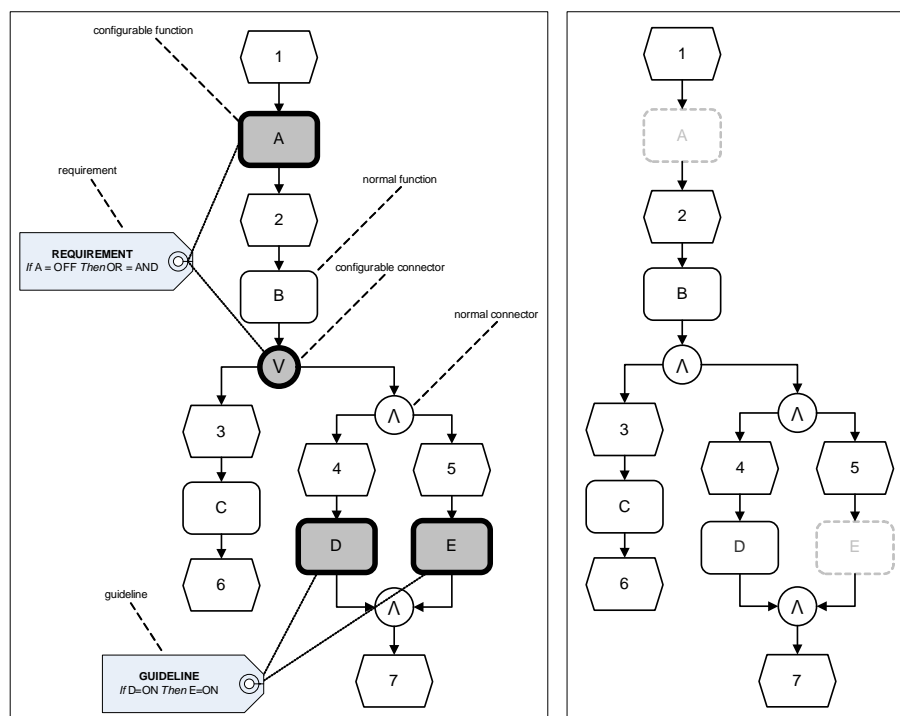


Figure 1: An example of a C-EPC (configurable and configured).

In a C-EPC functions and connectors can be configured. Notation-wise, these configurable nodes are denoted by bold line. *Configurable functions* may be included (ON), excluded (OFF), or conditionally skipped (OPT). To be more specific, for configurable functions, a decision has to be made whether to perform this function in every process instance at run-time, whether to exclude this function permanently, i.e. it will not be executed in any process instance, or whether to defer this decision to run time, i.e. for each process instance it has to be decided whether to execute the function or not. *Configurable connectors* subsume possible build-time connector types that are less or equally expressive. Hence, a configurable connector can only be configured to a connector type that restricts its behavior. A configurable OR-connector may be mapped to a regular OR-, XOR-, or

Table 1: Constraints for the configuration of connectors

Conf. Connector	Maps to <i>OR</i>	Maps to <i>XOR</i>	Maps to <i>AND</i>	Maps to <i>SEQ<sub>n</sub></i>
OR	X	X	X	X
XOR		X		X
AND			X	

AND-connector. Or, the OR-connector may be mapped to a single sequence of events and functions (indicated by  $SEQ_n$  for some process path starting with node  $n$ ). A configurable AND-connector may only be mapped to a regular AND-connector. A configurable XOR-connector may be mapped to a regular XOR-connector or to a single sequence  $SEQ_n$ . Table 1 illustrates the mapping constraints for configurable connectors.

In order to depict inter-dependencies between configurable EPC nodes, the concept of *configuration requirements* has been introduced. Inter-related configuration nodes may be constrained by such requirements. Consider the example given in the left part of Figure 1. If the configurable function A is excluded, the inter-related configurable OR-connector must be mapped to an AND-connector. Such configuration requirements are best defined via logical expressions in the form of *If-Then*-statements. Moreover, *configuration guidelines* provide input in terms of recommendations and proposed best practices (also in the form of logical *If-Then*-expressions) in order to support the configuration process semantically. Consider again the example given in the left part of Figure 1. A recommendation could be that if function D is included, then so should be function E. Summarizing, requirements and guidelines represent hard (*must*) respectively soft (*should*) constraints.

## 2.2 An Introduction to EPML

The specification of the EPC Markup Language (EPML) has been motivated by the heterogeneity of proprietary interchange formats of business process modeling tools [MN04, MN05]. It has been defined as a tool-neutral interchange format, following the design principles readability, extensibility, tool orientation, and syntactical correctness. EPML captures the essential concepts of EPCs. These include *function* and *event* type elements, *and*, *or*, and *xor* join and split connectors, as well as decomposition via *hierarchical functions* and *process interfaces*. The control flow is defined by *arcs* between these elements. Beyond that, EPML introduces concepts that leverage the interchange between process repositories. An EPML file can store multiple EPC models. These can be arranged in a hierarchy of *directories* that allows for a logical grouping of models. So-called *definitions* allow to define a logical model element that is used several times in a graphical model.

The elements of an EPC are grouped into an `epc` which is identified by a unique `epcId`. Each EPC element is represented as a subelement of `epc` with the element name indicating its EPC element type, i.e. functions are given as `function` elements, events as `event` elements etc. Each EPC element is assigned a unique `id` of integer type and a

name. The `arc` elements define the control flow via their `flow child` element. Each arc is pointing from the EPC element whose `id` is referenced in the `source` attribute to that element referenced in the `target` attribute. All elements of an EPC may include graphical information about their layout, position, and size in the model. Additional information can be annotated to process elements via `attributes`. Each attribute has to reference an attribute type, which helps to identify attributes of the same kind. Furthermore, relationships between control flow elements and `participant`, `dataField`, as well as `application` elements can be defined. Due to space restrictions we abstain from an in-depth discussion of the EPML here. For details refer to [MN05].

### 2.3 Requirements for a C-EPC Representation in EPML

A representation of C-EPCs in EPML demands a formal definition of the C-EPC as a basis for specifying an interchange format. The definition of the C-EPC syntax is based on the notion of classical EPCs as defined in [KNS92] and formally specified in [vdA99]. For more information on formal descriptions of EPCs refer to [MN05]. For the purpose of this paper we stick to the C-EPC formalization given in [RvdA05].

**Definition 1 (Configurable EPC)** A *Configurable EPC (C-EPC)* is a ten-tuple defined as  $(E, F, C, l, A, F^C, C^C, O^C, R^C, G^C)$  such that:

- $E, F, C, l$ , and  $A$  refer to standard EPC elements *events*, *functions*, *connectors*, a mapping to define a *label* AND, OR, or XOR for each connector, and *arcs* [vdA99].
- $F^C \subseteq F$  is the set of *configurable functions*.
- $C^C \subseteq C$  is the set of *configurable connectors*.
- $O^C \subseteq (F^C \cup C^C) \times (F^C \cup C^C)$  is a *partial order* over the configuration nodes defining the suggested order in which these nodes are mapped to concrete values.
- $R^C$  is the set of *configuration requirements*.
- $G^C$  is the set of *configuration guidelines*.

Both  $R^C$  and  $G^C$  contain statements about binding configurable nodes to concrete values. Consider the example of a requirement like “if  $c \in C^C = XOR$  then  $f \in F^C = ON$ ”. This represents a logical predicate about a configuration that must hold true. In contrast to that, a guideline represents an advise that may or may not be fulfilled by a configuration. Accordingly, the latter also represent predicates, but these do not necessarily hold true. Nevertheless, a logic-based representation of guidelines provides for formal evaluation in how far a configuration complies with industry best practices given in the guideline.

Concerning configurable connectors, we formalize the constraints given in Table 1 by a partial order. This *partial order*  $\leq^C$  specifies which concrete connector type may be used for a given configurable connector type.

**Definition 2 (Partial Order)** The *partial order*  $\leq^C$  is defined on  $CT = \{AND, OR, XOR\} \cup CTS$  where  $CT$  is the set of connector types and the sequence operator is referred to as  $CTS = \{SEQ_n | n \in E \cup F \cup C\}$ .

According to Table 1 the partial order is defined as  $\leq^C = \{(AND, AND), (XOR, XOR), (OR, OR), (XOR, OR), (AND, OR)\} \cup \{(n, XOR) | n \in CTS\} \cup \{(n, OR) | n \in CTS\} \cup \{(n, n) | n \in CTS\}$ .

For example,  $XOR \leq^C OR$  and  $SEQ_n \leq^C XOR$  imply that the second configurable connector can be mapped to the first connector or a sequence, respectively.

Before specifying a configuration, we need to define the notation for sets of predecessor and successor nodes.

**Notation 1 (Predecessor and Successor Nodes)** Let  $N$  be a set of *nodes* and  $A \subseteq N \times N$  a binary relation over  $N$  defining the arcs. For each *node*  $n \in N$ , we define the set of *predecessor nodes*  $\bullet n = \{x \in N | (x, n) \in A\}$ , and the set of *successor nodes*  $n \bullet = \{x \in N | (n, x) \in A\}$ .

Then, a configuration maps all configurable nodes to concrete values.

**Definition 3 (Configuration  $l^C$ )** Let C-EPC =  $(E, F, C, l, A, F^C, C^C, O^C, R^C, G^C)$  be a C-EPC. The mapping  $l^C \in (F^C \rightarrow \{ON, OFF, OPT\}) \cup (C^C \rightarrow CT)$  is called a *configuration of C-EPC* if for each  $c \in C^C$ :

- $l^C(c) \leq^C l(c)$ .
- If  $l^C(c) \in CTS$  and  $c \in C_J$ , then there exists an  $n \in \bullet c$  such that  $l^C(c) = SEQ_n$ .
- If  $l^C(c) \in CTS$  and  $c \in C_S$ , then there exists an  $n \in c \bullet$  such that  $l^C(c) = SEQ_n$ .

The function  $l^C$  maps configurable functions onto concrete values. Configurable connectors are mapped onto the set  $CT$ . Clearly this mapping should be consistent with the constraints depicted in Table 1 and the partial order  $\leq^C$ . Moreover, if  $l^C(c) = SEQ_n$ , then  $n$  should be in the pre-set (for a join connector, i.e.  $c \in C_J$ ) or post-set (for a split connector, i.e.  $c \in C_S$ ) of  $c$ .

The right part of Figure 1 shows an example for a configured C-EPC model where the configurable OR-connector has been mapped to a regular AND-connector and where function A and D have been excluded. What, however, hasn't been ensured at that point is that the configured C-EPC is depicted as a traditional lawful EPC. As outlined in [RRvdAM05], this translation process bears in itself semantic and syntactical complications, which result in a need for tool support based on a rigorous formal specification. Hence, in the following, we take the formal syntax of C-EPCs presented here as a starting point to specify a schema for C-EPCs via an extension to EPML. This may then be used to validate the syntactical correctness of both C-EPCs and configurations of C-EPCs, respectively.

### 3 Towards an EPML Schema for C-EPCs

#### 3.1 The C-EPML Schema

Forthcoming from our preceding elaborations on the two statuses of C-EPCs (see Section 2.1), we must seek a specification that allows for a representation of C-EPCs both *before* configuration, i.e. configurable C-EPCs, and *after* configuration, i.e. configured C-EPCs, in accordance to the definitions specified in the previous section.

New EPML elements need to be defined. These elements have to capture the C-EPC extensions to EPCs, i.e. configurable functions, configurable connectors, a partial order of configuration nodes representing a suggested order of configuration, configuration requirements, and configuration guidelines. We base our C-EPC extension of EPML on the principle of *EPML compatibility*: on the one hand, standard EPC models represented in EPML should still be valid against the C-EPC extended EPML schema. On the other hand, EPML tools that are not aware of configuration aspects should still be able to process a C-EPC as a standard EPC, simply by ignoring additional configuration information. This implies that we do not introduce additional EPML elements for configurable functions and connectors but instead define attribute elements to annotate `epc`, `function`, and `connector` elements with configuration information in a structured manner. Also, we need to distinguish between configurable and configured C-EPCs. We thus introduce further `configuration` attributes annotated to configurable functions and connectors that optionally capture the desired configuration values.

**Configurable Functions:** Configurable functions  $f \in F^C$  are specified in EPML as `function` elements having a respective `configurableFunction` child element. Those functions not having this child element are standard EPC functions that cannot be configured. In order to depict that the function has been configured, the configuration value has to be given in a further nested `configuration` element and its `value` attribute. This attribute is restricted to the values `on`, `off`, and `opt`. The XML syntax is as follows (question marks indicate that an element is optional):

```
<function id = 'xs:integer'>
  <configurableFunction> ?
    <configuration value = 'on|off|opt' /> ?
  </configurableFunction>
</function>
```

**Configurable Connectors:** Configurable connectors  $c \in C^C$  are specified as EPML `and`, `or`, or `xor` elements having a respective `configurableConnector` child element. Those connector elements not having this child element are standard EPC connectors that cannot be configured. In order to depict that the connector has been configured, the configuration value has to be given in a further nested `configuration` element and its `value` attribute. According to Table 1, this attribute is restricted to the values `and` for

and connectors; `seq`, `and`, `xor`, and `or` for `or` connectors; and `seq` and `xor` for `xor` connectors. The XML syntax is as follows, e.g. for an `or` connector:

```
<or id = 'xs:integer'>
  <configurableConnector> ?
    <configuration value = 'seq|and|xor|or'
      goto = 'xs:integer' ? /> ?
  </configurableFunction>
</or>
```

Note that for configurable connectors that may be mapped to a single sequence, a conditional `goto` attribute is specified that serves as a pointer to a successor node of the sequence that is selected. Using XML schema, it is not possible to check the correctness of such sequence configuration. We will address this problem in Section 3.2.

**Partial Order of Configuration Nodes:** The order of configuration  $o \in O^C$  is specified using standard EPML relation elements. The type of a relation has to be specified as a definition in the header of an EPML file. The partial order of configuration must be defined with a `defId` set to `ConfOrder`. In an `epc` element the `from` and `to` attributes of a relation define a “first `from`, then `to`” order relationship between the referenced elements. The XML syntax is as follows:

```
<definition defId = 'ConfOrder' />
...
<relation id = 'xs:integer'
  from = 'epml:refFromId'
  to   = 'epml:refToId' />
```

Note that this partial order merely specifies a suggested order of configuration, hence it denotes a predicate for configuration that may or may not hold true. Indeed, it is an optional element that may be used to streamline and guide the process of model configuration.

**Configuration Requirements:** The configuration requirements  $r \in R^C$  are represented by child elements of the respective `epc` process container element. The `idRefs` attribute holds a list of references to those elements that are related to the requirement. The requirement is composed of a `if` and a `then` part. Both contain `xpath` attributes holding XPath statements (for further details on XPath refer to [CD99]). These statements are evaluated either *relative* to the `epc` node or *absolute* against the EPML document. Using absolute statements, constraints can be defined across several C-EPC models. If the `if` part of the requirement is true, the `then` part must be true, too. The XML syntax is as follows:

```
<configurationRequirement idRefs = 'list of xs:integer'>
  <if xpath = 'xpath-statement' />
  <then xpath = 'xpath-statement' />
</configurationRequirement>
```

**Configuration Guidelines:** Configuration guidelines  $g \in G^C$  are captured by the same XML structure as requirements. Still, there is the semantic difference that requirements *must* be met, and guidelines *should* be met. The XML syntax is as follows:

```
<configurationGuideline idRefs = 'list of xs:integer'>
  <if xpath = 'xpath-statement' />
  <then xpath = 'xpath-statement' />
</configurationGuideline>
```

As a summary of the EPML extension for C-EPCs, we pick up the example from Figure 1. Figure 2 gives the same C-EPC model with its EPML representation. The configurable functions, configurable connectors and the configuration order are easy to identify. Note that the `id` attributes hold arbitrary integer values. We will now briefly explain the XPath statements of the configuration requirement. The `if` statement of the requirement identifies a function element. As defined above, a relative statement is evaluated in the context of the current `epc` element. This means here that `function` elements of other `epc` elements of the EPML file are not considered. The square brackets define a constraint in XPath. Accordingly, the requirement defines that if a function with `id = 7` and configuration mapped to `off` is part of the EPC, then an OR connector with `id = 9` and which is configured to AND must exist in the model, too. The configuration guideline works similarly.

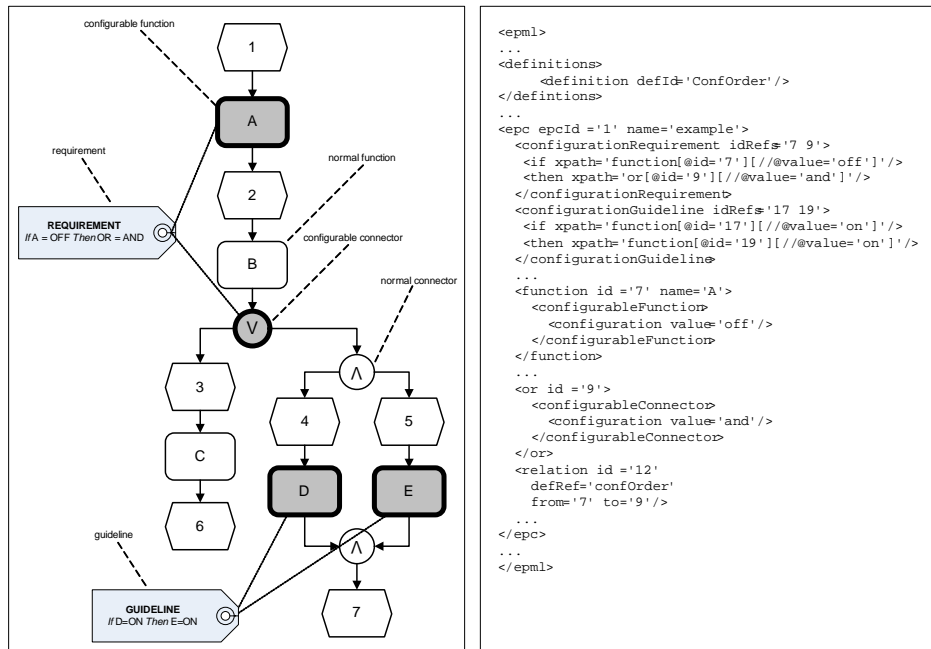


Figure 2: An example of a C-EPC and parts of its EPML representation.

### 3.2 On the Automatic Validation of C-EPCs

Building on the EPML-based interchange format for C-EPCs, various tools can be developed for the support of reference model configuration. As a first step towards comprehensive tool support for C-EPCs, we have implemented an C-EPC validation program called *C-EPC Validator*<sup>4</sup>. The C-EPC Validator dynamically evaluates the XPath statements given in configuration requirements and guidelines to assess the validity of selected model configurations. As XML schema languages do not support complex constraints as defined by C-EPC's configuration requirements and guidelines (for more information on the limitations of XML schema languages in the context on EPCs refer to [MN03]), we chose an XSLT-based implementation. Furthermore, to allow for dynamic evaluation of XPath statements we use EXSLT extensions<sup>5</sup>.

The C-EPC Validator takes an EPML document as input and generates a report as an HTML document. Basically, three kinds of properties are validated. First, the C-EPC Validator checks the syntactical correctness of the C-EPC. This is partially redundant to the validation that can be done using the EPML schema. Yet, on the schema level it is not possible to check whether connectors are correctly configured to sequences. A configuration value of `seq` requires the optional `goto` attribute to be included. Furthermore, this attribute must point to a successor node of the connector. This constraint is specified using XSLT. Second, the C-EPC Validator checks the compliance of the configured model to the requirements. Conditions and implications are highlighted with green and red font color to easily identify violations. As an example consider Figure 3. The left part shows a possible configuration of the C-EPC presented in the left part of Figure 1. As can be seen, the configuration requirement is violated as the configurable *OR*-connector has been mapped to an *XOR*-connector although the configurable function *A* has been mapped to `off`. Third, the same mechanism as for requirements is used to validate the guidelines. Violations are given in orange font color.

## 4 Related Work

### 4.1 Configurable Reference Models

Research on configurable reference models can be divided into requirements engineering for the development of Enterprise Systems, e.g. [Br99], and requirements engineering for the configuration of Enterprise Systems. The latter is the focus of this paper.

Related work on configurative reference modeling includes the perspectives-based configurative reference process modeling approach by Becker et al. [BDD<sup>+</sup>04]. This approach proposes several mechanisms for automatically transforming a reference model into an

<sup>4</sup>The C-EPC Validator and the EPML Schema that supports C-EPCs can be downloaded from <http://wi.wu-wien.ac.at/~mending/EPML>

<sup>5</sup>The program can be run using XSLT processors that understand EXSLT extensions, e.g. the XALAN processor (see <http://xml.apache.org/xalan-j>). For EXSLT extensions to XSLT see <http://www.exslt.org>.

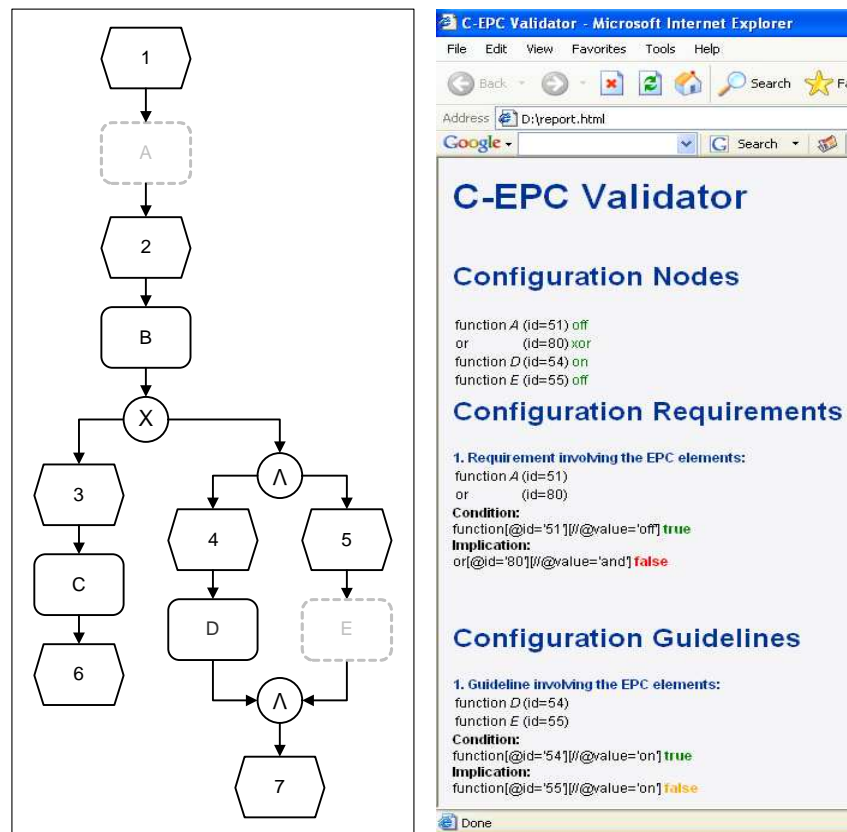


Figure 3: A configured C-EPC, and the validation report of the C-EPC Validator.

individual model. While the work of Becker et al. focuses on generic adaptation mechanisms, our research pursues a reference model-driven approach towards ES configuration and moreover considers the technical stage of model translation as well. Soffer et al.'s suggestions on ERP modeling [SGD03] can also be regarded as close to our proposed ideas. Following the concept of scenario-based requirements engineering, they evaluate the Object-Process Modeling Methodology in order to determine a most appropriate ERP system representation language. The so-called argumentation facet, related to the ability of a modeling language to express optionality-related information, is one of their many criteria. Their work does not comprehensively analyze requirements related to modeling ERP configurability and focuses on technique evaluation rather than on the development of a more appropriate technique. Brehm et al. discuss alternative ways of configuring Enterprise Systems [BHM01]. Their taxonomy for ERP configuration and customizing is widely cited. However, they do not demonstrate how this work can be linked to reference models of Enterprise Systems.

## 4.2 Business Process Modeling Interchange Formats

Various interchange formats and schemas have been proposed for business process modeling. For an overview refer to [MNN04]. Among the design criteria for interchange formats, the criterion of extensibility (cf. e.g. [Ko92]) is of special importance to our work. As we extend EPML with C-EPC concepts, it has to be granted that documents complying with old versions of the schema remain valid. Our approach is to add optional elements nested in `epc`, `function`, and `connector` elements.

Among interchange formats that have been proposed for business process modeling, none captures configuration directly. In terms of flexible generation of process models, the idea behind OWL-Services (OWL-S) [APS<sup>+</sup>03] is closest to our approach. OWL-S is a service meta model represented in OWL building on an (input-output-preconditions-effects) quadruple to describe services. OWL-S is aimed at languages to enable dynamic composition of processes. Yet, such a dynamically composed process is generated via automatic reasoning on service descriptions stored in a repository, and not by manual configuration as in the case of C-EPCs. Furthermore, dynamically built service compositions are not explicitly meant for reuse as a process model, but rather for one-time instantiation. Other languages like Business Process Execution Language for Web Services (BPEL4WS, WS-BPEL, or BPEL) [ACD<sup>+</sup>03] provide some flexibility as they permit dynamic identification of service endpoints. This may also be considered as related to configuration aspects in general. Yet, these aspects are not standardized in the specification. Furthermore, BPEL is directed at the definition of executable processes rather than configurable conceptual process models.

As far as we are aware, C-EPC is the only dedicated Configurable Business Process Modeling language and EPML the only BPM interchange format that support configuration aspects on a conceptual level.

## 5 Conclusion and Future Work

This paper presented an XML schema-based specification of the C-EPC using the EPML format. We reported on the design of the C-EPC compliant schema and outlined a prototype called C-EPC Validator for the validation of model configuration using the C-EPC.

Our research has a few limitations. First, our conceptual approach needs to be empirically validated with business practitioners to prove its feasibility and applicability. This task is currently being conducted. Second, we have not yet approached the task of translating C-EPCs to other process specifications. However, as the schema is now available, we are currently working on transformations and respective tool support. Third, there is not yet adequate tool support for the modeling of C-EPCs. However, transformations between EPML and ARIS Markup Language are available. Furthermore, the SemTalk business process modeling tool and the open-source modeling platform EPCTools support EPML. Hence, existing modeling solutions can easily be modified and adapted to cater for the C-EPC extensions of the EPC modeling language. Again, this task will be approached in

the near future.

**Acknowledgements.** The authors would like to express their gratitude towards the continuous fruitful contributions of Markus Nüttgens to the EPML initiative. The research on the C-EPC has been funded by SAP Research and Queensland University of Technology with the ARC Linkage project “Modelling Configurable Business Processes”. SAP is a trademark of SAP AG, Germany.

## References

- [ACD<sup>+</sup>03] Andrews, T., Curbera, F., Dholakia, H., Golland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., und Weerawarana, S.: Business Process Execution Language for Web Services, Version 1.1. Specification. BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems. 2003.
- [APS<sup>+</sup>03] Ankolenkar, A., Paolucci, M., Srinivasan, N., Sycara, K., Solanki, M., Lassila, O., McGuinness, D., Denker, G., Martin, D., Parsia, B., Sirin, E., Payne, T., McIlraith, S., Hobbs, J., Sabou, M., und McDermott, D.: OWL-S: Semantic Markup for Web Services (Version 1.0). Specification. OWL Services Coalition. 2003.
- [BDD<sup>+</sup>04] Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., und Kuroпка, D.: Configurative Process Modeling - Outlining an Approach to increased Business Process Model Usability. In: Khosrow-Pour, M. (Hrsg.), *14th Information Resources Management Association International Conference*. S. 615–619. New Orleans. 2004. IRM Press.
- [BHM01] Brehm, L., Heinzl, A., und Markus, M. L.: Tailoring ERP Systems: A Spectrum of Choices and their Implications. In: Nunamaker Jr, J. F. und Sprague, R. H. J. (Hrsg.), *34th Hawaii International Conference on System Sciences*. S. 8017. Maui. 2001. IEEE.
- [Br99] Brinkkemper, S.: Requirements Engineering for ERP: Requirements Management for the Development of Packaged Software. In: *4th International Symposium on Requirements Engineering*. S. 159. Limerick, Ireland. 1999.
- [CD99] Clark, J. und DeRose, S.: XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November. World Wide Web Consortium. 1999.
- [CKL97] Curran, T., Keller, G., und Ladd, A.: *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Enterprise Resource Planning Series. Prentice Hall PTR. Upper Saddle River. 1997.
- [Da98] Davenport, T. H.: Putting the Enterprise into the Enterprise System. *Harvard Business Review*. 76(4):121–131. 1998.
- [HC93] Hammer, M. und Champy, J.: *Reengineering the Corporation: A Manifesto for Business Revolution*. Harpercollins. New York. 1993.
- [KNS92] Keller, G., Nüttgens, M., und Scheer, A. W.: Semantische Prozessmodellierung auf der Grundlage “Ereignisgesteuerter Prozessketten (EPK)”. Technical Report 89. Institut für Wirtschaftsinformatik Saarbrücken. Saarbrücken, Germany. 1992.
- [Ko92] Koegel, J. F.: On the Design of Multimedia Interchange Formats. In: *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, 12-13 November 1992, La Jolla, California, United States*. S. 262–271. 1992.

- [LLO93] Luftman, J. N., Lewis, P. R., und Oldach, S. H.: Transforming the Enterprise: The Alignment of Business and Information Technology Strategies. *IBM Systems Journal*. 32(1):198–221. 1993.
- [MN03] Mending, J. und Nüttgens, M.: EPC Syntax Validation with XML Schema Languages. In: M. Nüttgens and F. J. Rump (Hrsg.), *Proc. of the 2nd GI-Workshop on Business Process Management with Event-Driven Process Chains (EPK 2003)*, Bamberg, Germany. S. 19–30. 2003.
- [MN04] Mending, J. und Nüttgens, M.: Exchanging EPC Business Process Models with EPML. In: Nüttgens, M. und Mending, J. (Hrsg.), *XML4BPM 2004, Proceedings of the 1st GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 7th GI Conference Modellierung 2004, Marburg Germany*. S. 61–80. <http://wi.wu-wien.ac.at/~mending/XML4BPM/xml4bpm-2004-proceedings-epml.pdf>. March 2004.
- [MN05] Mending, J. und Nüttgens, M.: EPC Markup Language (EPML) - An XML-Based Interchange Format for Event-Driven Process Chains (EPC). Technical Report JM-2005-03-10. Vienna University of Economics and Business Administration, Austria. 2005.
- [MNN04] Mending, J., Nüttgens, M., und Neumann, G.: A Comparison of XML Interchange Formats for Business Process Modelling. In: Feltz, F., Oberweis, A., und Otjacques, B. (Hrsg.), *Proceedings of EMISA 2004 - Information Systems in E-Business and E-Government*. volume 56 of *Lecture Notes in Informatics*. 2004.
- [RRvdAM05] Recker, J., Rosemann, M., van der Aalst, W., und Mending, J.: On the Syntax of Reference Model Configuration - Transforming the C-EPC into Lawful EPC Models. In: Kindler, E. und Nüttgens, M. (Hrsg.), *First International Workshop on Business Process Reference Models (BPRM'05)*. S. 60–75. Nancy, France. September 2005.
- [RvdA05] Rosemann, M. und van der Aalst, W.: A Configurable Reference Modelling Language. *Information Systems*. to appear. 2005.
- [Sc00] Scheer, A. W.: *ARIS business process modelling*. Springer Verlag. 2000.
- [SGD03] Soffer, P., Golany, B., und Dori, D.: ERP modeling: a comprehensive approach. *Information Systems*. 28(6):673–690. 2003.
- [vdA99] van der Aalst, W. M. P.: Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*. 41(10):639–650. 1999.