



**Queensland University of Technology**  
Brisbane Australia

This is the author's version of a work that was submitted/accepted for publication in the following source:

Recker, Jan C., Rosemann, Michael, van der Aalst, Wil M. P., & Mendling, Jan (2005) On the syntax of reference model configuration: Transforming the C-EPC into lawful EPC models. In Kindler, Ekkart & Nüttgens, Markus (Eds.) *First International Workshop on Business Process Reference Models (BPRM'05)*, 5 September, Nancy, France.

This file was downloaded from: <http://eprints.qut.edu.au/2093/>

© Copyright 2005 (please consult author)

**Notice:** *Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source:*

# On the Syntax of Reference Model Configuration – Transforming the C-EPC into Lawful EPC Models<sup>1</sup>

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

<sup>1)</sup> Faculty of Information Technology  
Queensland University of Technology  
126 Margaret Street, Brisbane QLD 4000, Australia  
{j.recker, m.rosemann}@qut.edu.au

<sup>2)</sup> Eindhoven University of Technology  
P.O. Box 513, 5600 MB Eindhoven, The Netherlands  
w.m.p.v.d.aalst@tm.tue.nl

<sup>3)</sup> Department of Information Systems and New Media  
Vienna University of Economics and Business Administration  
A-1180 Vienna, Austria  
jan.mendling@wu-wien.ac.at

**Abstract.** For Enterprise Systems (ES) to provide support for the business operations of enterprises, they need to be *configured* to fit organizational requirements. ES reference models aim at supporting this task but, up to now, fail in providing adequate conceptual support due to missing configurability of the models themselves. This paper extends the work on *a configurable reference modeling notation*. In previous research we developed an adequate conceptual notation for configurable reference models. This paper considers a *syntactic* perspective of reference model configuration. We discuss the lawful environments of configurable nodes and report about syntactic and semantic implications of model configuration in these environments. We then apply these findings in the *design of a XML-based interchange format for reference model configuration* and discuss its applicability for the *conceptual design of tool support* for the configuration of reference process models, which will facilitate and aid the verification of the syntactical correctness of configured reference process models that may then be mapped to executable process specifications.

**Keywords.** Process configuration, reference modeling, Enterprise Systems

## 1 Introduction – Reference Models and Enterprise Systems

Many organizations suffer problems from badly implemented Enterprise Systems (ES) [1]. Both academia and industry state that these problems result from a misalignment gap between business and IT, which, once closed, would lead to signifi-

---

<sup>1</sup> The research on the C-EPC is financially supported by SAP Research.

cantly improved business performance [2]. The notion of (mis-) alignment primarily embraces the process dimension, *i.e.* the alignment of IT functionality to the actual business processes of an organization. In many cases, it is observed that the system hampers the normal way of handling processes instead of supporting it.

This is even more 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 [3, 4]. Alongside this trend, the IS community has experienced the proliferation of an enormous number of process modeling methods, including the Event-Driven Process Chains (EPC) [5], which itself is used within the Enterprise System SAP.

The term Enterprise Systems represents integrated information systems that aim at holistically supporting the operational processes of organizations. Though ES packages are distributed as commercial off-the-shelf (COTS) software, their implementation often results in tremendous configuration efforts. Given the fact that the alignment of "generic" ES solutions to "specific" organization needs denotes a highly complex task, it was found that a model-driven solution would provide a more intuitive approach towards configuring, adapting, and customizing ES software to customer demands. Such a model-driven approach naturally would take on existing ES *reference models*, which have already been developed by ES vendors in order to improve the understandability of their systems.

In the context of Enterprise Systems, such *application reference models*, describing the structure and functionality of software solutions on different levels of conceptual abstraction [6], are of particular interest. Due to their prescriptive nature, *i.e.* application reference models usually depict the complete functionality of the system [7], they are, however, only of limited use to the ES configuration process, mainly due to a lack of conceptual support in the form of a configurable modeling language underlying the reference models.

Addressing this issue, we have been developing a new reference modeling approach which considers the configurable nature of an Enterprise System. The representation language of this approach is called a *Configurable EPC* (C-EPC). While previous research efforts have been focused on the meta model and the notation of C-EPCs [8], this paper discusses syntactical problems of C-EPCs in the light of reference model configuration. The *scope of our paper* is the translation of (configured) C-EPC models into lawful (regular) EPCs. We will show that the application of C-EPC in the process of ES reference model configuration leads to syntactic problems and we will outline an approach how to handle these problems when translating C-EPC models into lawful process models. More specifically, the *aim of our paper* is to outline a XML schema-based approach using the EPC Markup Language (EPML) [9] for the task of syntactical validation of reference process model configuration.

The remainder of our paper is structured as follows: Section 2 presents issues and shortcomings of the EPC notation in the light of reference model configurability and introduces the notion of a configurable reference process modeling technique. Also, it briefly reports on related work in the field of configurative reference modeling. Section 3 discusses semantic and syntactic problems that occur when configuring reference process models. We then present a XML-based specification of C-EPCs on which the design of tool support for syntax validation and automatic model transla-

tion will be based. We briefly summarize in Section 4 and propose conclusions drawn from our work.

## 2 A Configurable Reference Modeling Language

### 2.1 On syntax and semantics of EPCs

In order to gain an understanding for the C-EPC notation and to raise awareness of problems we encounter during reference process model translation, we briefly outline the notion of classical EPC models and discuss some issues related to the informal semantics and syntax of EPC.

The EPC language was developed at the University of Saarland, Germany, in collaboration with SAP AG (see [5]). A simple EPC consists of events as passive states, functions as active transformations, and logical connectors that connect events and states through control flow. EPCs have – amongst others – been used for the design of the reference process models in SAP [7].

As discussed quite intensively in academia, see *e.g.* [10-12], the definition of EPC in [5], on which we based our research on the C-EPC language, leads to syntactic and semantic problems. The syntax of EPCs as deployed in our research context can be found in [8]. However, this definition does not cover behavioral aspects of EPCs and thus may contain semantic ambiguities.

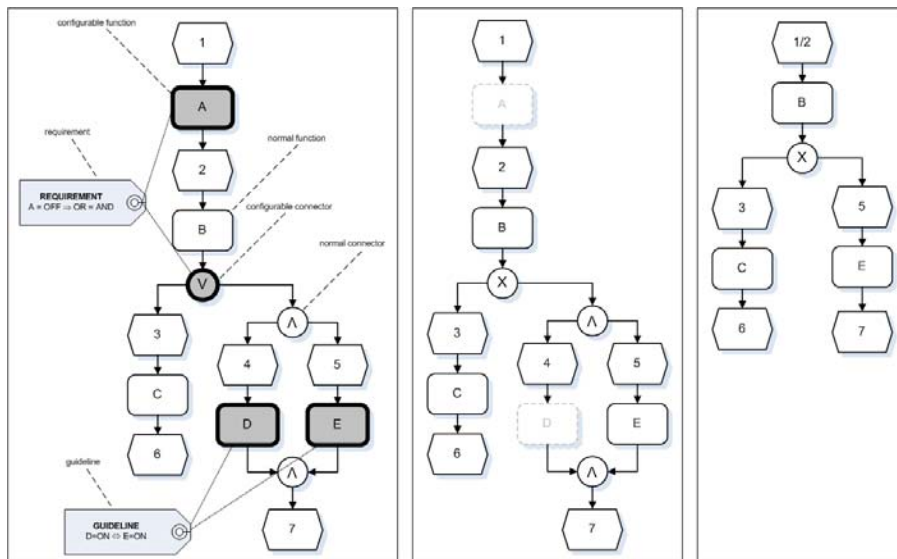
For instance, the informal semantics of an OR-join causes confusion as a joining OR-connector may or may not synchronize incoming process flows [10]. While these problems have been approached by academic contributions, see *e.g.* [11, 13, 14], in general the issue of informal semantics of EPCs must at this point be considered an open issue.

Considering such problems in the light of ES configuration, the informal semantics of EPC lead to severe issues: EPC models, which depict those process scenarios that are deemed relevant to a particular organization, need to be translated into executable process specifications, which an Enterprise System can execute at run-time. Or, consider a workflow management system that defines, executes, manages, and controls the business processes based on these models. In whatever case, it is of paramount importance to have syntactically correct, *i.e.* lawful EPC process models as an outcome of the configuration process that may then be translated.

In our research, however, we did not want to further complicate the semantics of (configurable) EPCs but decided to express the semantics of C-EPCs in terms of traditional EPCs. Hence, we seek to validate the behavior of configurable processes through their translation to regular EPCs. Then, any of the formalization approaches mentioned in [11, 13, 14] may be used as a semantic foundation, and we may stop the semantics discussion here. However, later we will need to discuss some semantic implications when translating Configurable EPCs into lawful process models.

## 2.2 On Configurable Reference Process Models: The C-EPC Notation

Current reference modeling languages lack configuration support. As an example, the SAP reference model [7], which is depicted in the EPC notation, covers in the version 4.6 more than 1,000 business processes and inter-organizational business scenarios. As the main objective of reference models is to streamline the design of particular models, they are coined by the “Design by Reuse” paradigm. To increase their applicability, such models typically not include merely one proposed alternative for conducting business in a certain domain but a range of often mutually exclusive alternatives. Hence it denotes an ‘upperbound’ of process models that may possibly be implemented in a particular enterprise. As an organization might merely favor one of the depicted alternatives, they potentially only refer to a subset of ES functionality to be implemented and accordingly only to a subset of the reference model. Up to today, however, these types of decision cannot be reflected within the ‘upperbound’ reference model due to lacking configuration support of the underlying reference modeling language. Existing reference modeling techniques do not support the highlighting and selection of different (process) configuration alternatives. This lack of expressiveness obviously denotes a major issue for reference model users.



**Fig. 1.** A simple C-EPC (before configuration, with selected configuration, and resulting EPC)

Addressing these issues, this section introduces *Configurable EPCs* (C-EPCs) (see Fig. 1) as an extension to the popular EPC modeling technique. Focus was spent to the active parts of process models, *i.e.* functionality (functions, tasks, transitions, and the like) and control flow. We have not examined the configurability of events (or states) as more passive parts of processes since they cannot actively be influenced by an organization. It is the reaction to events that can be influenced and this reaction is covered in C-EPCs. The notion of a configurable EPC has been introduced and for-

malized in [8], therefore we only discuss the basic notation here. Fig. 1 shows an example of a C-EPC model, with the left part showing the configuration alternatives, the middle part showing *one* selected configuration alternative, and the right part showing a lawful resulting EPC model based on that configuration.

In a C-EPC functions and connectors can be configured. Notation-wise, these configurable nodes are denoted by thick circles. *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 during run time, whether to dispose of this function permanently, *i.e.* it will not be executed in any process instance, or whether to defer this discussion to run time, *i.e.* for each process instance it has to be decided whether to execute the function or not.

*Configurable connectors* may be restricted. A configurable OR<sup>C</sup>-connector may be mapped to a regular OR-, XOR-, or AND-connector. Also, the OR-connector may be disposed of, resulting in a normal process sequence SEQ<sub>i</sub>. A configurable AND-connector may only be mapped to a regular AND-connector with a decision being made as to how many of  $n$  available process sequences are to be executed in synchronization. A configurable XOR-connector may be mapped to a regular XOR-connector with a decision being made as to how many of  $n$  available mutually exclusive process paths are to be provided for execution, or the XOR-connector may be disposed of, resulting in a single process sequence SEQ<sub>i</sub>.

In order to depict inter-dependencies between configurable EPC nodes, the notion of *configuration requirements* has been introduced. Inter-related configuration nodes may be limited by these requirements (constraints, expressed denoted by logical expressions). Consider the example given in Fig. 1. If the configurable function A is excluded, the inter-related configurable connector OR<sup>C</sup> must be mapped to a regular AND-connector.

Moreover, in order to provide input in terms of recommendations and proposed best practices, *configuration guidelines* may be depicted to guide the configuration process semantically. Consider again the example given in Fig. 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.

Concluding, we introduced a configurable reference modeling notation which potentially facilitates a model-driven selection and modification of process flows and process activities.

### 2.3 Related Work

Related work on configurative reference modeling includes the perspectives-based configurative reference process modeling approach by BECKER *et al.* [15]. This approach focuses on adaptation mechanisms and proposes several mechanisms for automatically transforming a reference model into an individual model. While the work of BECKER *et al.* focuses on generic adaptation mechanisms, this research pursues a reference model-driven approach towards ES configuration.

SOFFER *et al.*'s suggestions on ERP modeling [16] 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 just one of many of their 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.

GULLA and BRASETHVIK [17] introduce three process modeling tiers to manage the complexity of process modeling in comprehensive ERP Systems projects. Their functional tier dimension deals with the functionality of the Enterprise System. However, they do not study how reference models fit into in this tier.

### 3 On the Syntax of Reference Model Configuration

#### 3.1 Configuration using the C-EPC modeling language

The task of configuring reference models that have been deemed configurable by highlighting variation points embraces both a semantic and a syntactic dimension. While the former is concerned with making business configuration decisions in order to match organizational strategy and requirements, the latter is concerned with maintaining syntactical correctness within the configured models to ensure a lawful translation into executable systems at run time. We will show, that these dimensions are inter-related during configuration as syntactic considerations of implementing the models do have semantic, *i.e.* business consequences and must hence be considered during semantic configuration.

We have described the semantic dimension of configuration in [18]. Simplistically, through the use of the C-EPC notation, process scenarios and process alternatives that are deemed desirable for a particular organization are selected. This is done by switching configurable nodes within a C-EPC model to a desired setting. Configuration requirements and configuration guidelines restrict respectively aid this task.

The outcome of this phase is a C-EPC model where all configurable nodes have been switched to a certain setting. What, however, hasn't been ensured yet, is that these configured C-EPC models apply to the formal syntax of regular EPC. As an example, the middle part of Fig. 1 shows a configured version of the C-EPC model shown in the left part, where the configurable OR-connector has been switched to a regular XOR-connector and where function A and D have been excluded (shaded grey).

As can be seen, the resulting process model would be syntactically inconsistent: Consider function A: Assuming the control flow is reconnected where the excluded function is missing, two events would follow each other. This is syntactically incorrect.

Inadvertently, the step beyond semantic configuration of C-EPC models from a business perspective is the task of re-establishing syntactical correctness and consistency, *i.e.* the translation of configurable process model into lawful regular process specifications (as an example refer to the right part of Fig. 1).

### 3.2 Translating C-EPCs into EPCs: Syntactical and Semantic Problems

Now, in order to approach the syntactic and inherent semantic problems that arise due to the configuration of C-EPCs, we need to develop a translation approach that maps a configured C-EPC to a lawful regular EPC. This is a delicate task due to the semantic problems of EPCs themselves, as discussed above. There are in principle several options to approach this task:

- Refine the EPC specification to arrive at rigorously and unambiguously defined semantics for EPCs and thus, for C-EPCs.
- Ignore the semantics of EPCs and merely focus on specifying an unambiguous translation of C-EPCs to EPCs which themselves may then be further discussed.

Here, we opted for the latter alternative: We wanted to *extend* the work on reference modeling techniques rather than developing new ones. Due to its popularity for the design of reference models and referring to the extensive academic work on its formalization and definition we deemed it better to take EPCs as both starting and ending point for our design of configurable process models instead of proposing yet another semantic and syntactic definition of EPCs.

Now, looking at the configuration of reference process models, this task can be divided into *global* and *local* decisions, with the former being based on the general model context and which can be made without studying the individual process model. Local decisions on the other hand require an explicit study of the relevant (parts of) process models. Our forthcoming discussion is focusing on the *local* aspects of configuration. We do not deem it necessary to explicitly address global decisions for the following reasons: First, EPCs and thus C-EPCs can be hierarchically structured by decomposing single EPCs into more detailed sub-models. Analogously, each (C-) EPC may be generalized to a simpler EPC on a coarser level of detail. Hence, all contexts of configurable nodes may eventually be drilled up to the smallest possible local environment, as will be discussed below. Second, the notion of C-EPCs provides explicit representation for the depictions of inter-dependencies and inter-relationships between configurable nodes. Hence, global inter-relationships between processes depicted in separate process models may be expressed, thereby not needing an explicit addressing of a global process context. Third, as current practice shows (consider *e.g.* the configuration of the SAP system), the process of reference model configuration starts at a very coarse level of detail with industry sector-spanning process models (in the SAP context: collaborative business scenarios). At this stage, configuration refers to deleting dispensable processes from high-level process models. It can be seen as more of a scoping exercise in a pre-implementation stage. Hence, global configuration decisions merely are decisions as to the inclusion or exclusion of processes, the former of which then need to be *locally* configured.

Concluding, we argue that configured C-EPC models can be transferred into lawful EPC models in accordance to laws based on the *local* syntactic environment of configurable nodes. We must, for the purpose of this paper, limit some of the discussions to examples. A complete discussion of all local environments for configurable nodes and the entire resulting process model variants would exceed the length of this paper and is furthermore deemed unnecessary for making our argument.

### Configurable Functions

Firstly, we investigate the local environments of configurable functions. As an EPC consist of events (E), functions (F), and splitting (S) respectively joining (J) connectors, there are nine different local environments for a configurable function A (see Fig. 2).

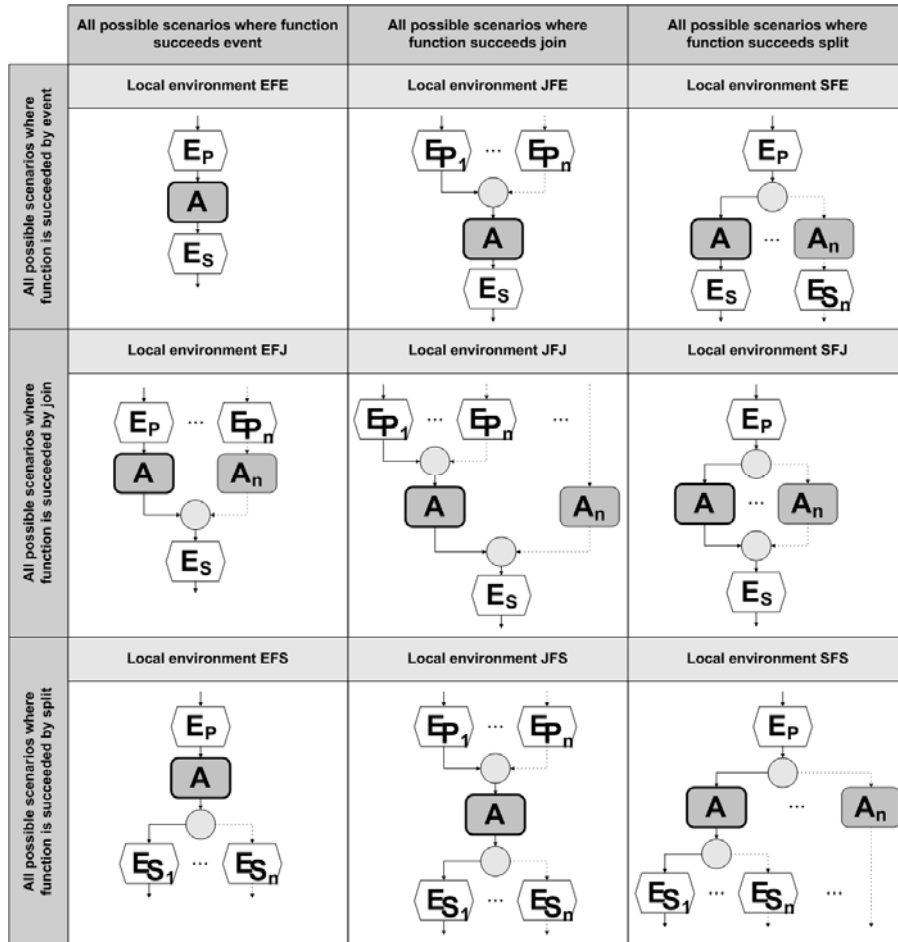


Fig. 2. Local environments for configurable functions

Studying the local environments of configurable functions it becomes obvious that, once a configurable function A has been switched to a desirable setting, the syntactical clean-up of the process model is not purely a technical decision. Due to missing formal semantics of the EPC notation – *e.g.* the EPC modeling language does not explicitly differ between triggering and resulting events that pre-/succeed a function – removals or inclusions of process model elements (such as the disposal of an event that follows a function) may have semantic and thus, business-related consequences.

Bearing that in mind, syntactic validation may lead to various syntactically lawful yet semantically different process models.

Consider the following example. Referring to the local environment ‘Event-Function-Event, EFE’ – the configurable function A is embedded in the context of a preceding event  $E_P$  and a succeeding event  $E_S$  – configuration and syntactic validation may lead to the process model variants shown in Fig. 3.

Now, as can be seen in Fig. 3, the syntactic handling of switching configurable functions “on” or “off” are simple, according to the definitions in [8]. Optional functions are trickier.

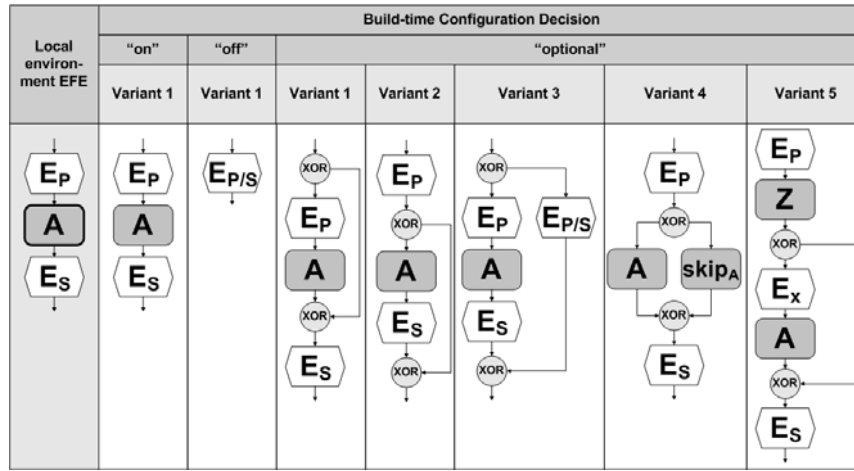


Fig. 3. Lawful alternatives for configuring a function in the EFE environment

Consider the configuration decision of switching the function A to “optional”. The resulting process model must cater for a run time decision to either bypass the function or execute it. Due to the informal EPC semantics, it is not necessarily obvious whether the succeeding event  $E_S$  denotes a triggering state for a subsequent business function or simply a resulting state for A. In the former case, the bypass does not need to include  $E_S$  (variant 1). In the latter case,  $E_P$  needs not to be bypassed (variant 2). Maybe both states surrounding A may be bypassed, thereby passing a new state  $E_{P/S}$  (variant 3). Another syntactically valid solution is to introduce a ‘dummy’ function “skipA” which just propagates a process folder from  $E_P$  to  $E_S$  without any transformation (variant 4). Or, a new decision function Z and an additional event  $E_x$  are introduced to augment the configuration decision of switching A to “optional” (variant 5). This case, obviously, requires the inclusion of knowledge external to the model in order to specify the decision function Z.

### Configurable connectors

Considering configurable connectors and referring back to the configuration constraints described in Section 2.2, these nodes may appear in any of the local environments shown in Fig. 4.

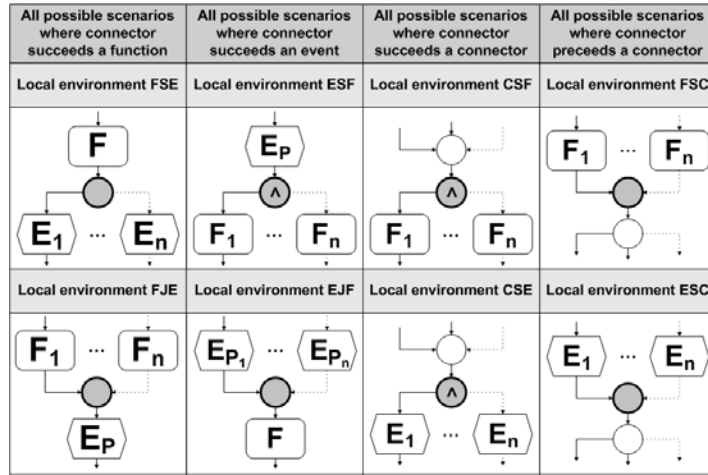


Fig. 4. Local environments for configurable connectors

According to the syntax rules of lawful EPC, some local environments are restricted to the AND connector, since both OR- and XOR-connector need to be linked to a preceding function that allows for the decision which branch to take. With respect to syntactically lawful process variants for these local environments, configurable connectors are easier to handle, as shown in Fig. 5.

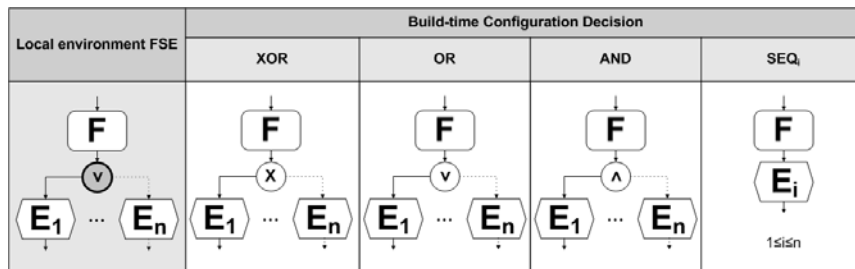


Fig. 5. Lawful alternatives for configuring an OR-connector in the FSE environment

As can be seen, for each configuration decision there exists exactly one syntactic lawful process variant. As can easily be shown, for each of the configurable XOR- and AND-connectors there exists merely one syntactic variant per desired setting as both configurable nodes may only be restricted in their behavior or mapped to a single sequence SEQ<sub>i</sub>. Analogously, as configurable connectors are defined to at most restrict their behavior, it is obvious that for each configuration in whatever local environment there can only exist one corresponding syntactically lawful process variant.

### Synopsis

The syntactic alternatives for all other local environments of configurable nodes, as depicted in Fig. 2 and Fig. 4 are constructed in a similar way. We examined the lawful environments of configurable nodes and constructed syntactic alternatives for all

combinations of predecessors and successors. As already mentioned, we cannot discuss them in detail here.

As can be shown through our examples, the syntactic clean-up of configured reference process models bears some semantic decisions in itself. Due to the inherent ambiguity of both syntax and semantics of the EPC modeling technique, syntactical validation of C-EPC models may lead to several syntactically lawful yet semantically different process model variants. Since we decided not to modify the EPCs but instead to base our work on the (arguably ambiguous) traditional EPC definition, there results a need for adequate tool support that facilitates and moreover aids the translation process from C-EPCs to EPCs. We will thus, in the next section, address this translation task by presenting a XML-based schema specification of C-EPCs that will be used to aid the syntax validation and translation of C-EPCs to regular lawful process models.

### 3.3 Towards Tool Support for Reference Model Configuration

Research towards tool support for C-EPCs based on an interchange format was motivated by two facts:

- A configuration of a C-EPC should correspond to a concrete EPC [8]. However, as we discussed in this paper, it is not possible to automate such mapping, hence adequate tool support is needed to facilitate and aid this task.
- EPCs and thus C-EPCs are not executable and thus cannot serve as specifications for process or workflow execution engines – which would, however, be desirable especially in the light of Enterprise Systems. In order to facilitate the interchange of configured reference process models to other process specifications, a standardized interchange format for “cutting-edge” process languages is needed.

Contemplating available options, we deemed a design specification based on a XML schema the best alternative. In particular, we opted for the *EPC Markup Language* (EPML) [9]. This selection was made for the following reasons: First, the EPML is able to perform syntax validations of EPCs [19]. Second, the EPML leverages the interchange of EPCs to other process modeling and execution languages [20], *e.g.* Petri nets. Third, EPML can be generated from the ARIS Markup Language (AML) and is also supported by open source modeling solutions [21], *e.g.* EPC Tools; hence, tool platforms are available for implementing reference model configuration tool support based on C-EPCs.

Now, due to space limitations we cannot give a thorough introduction to the EPML definition, which can be found at <http://wi.wu-wien.ac.at/~mending/EPML/>. Instead, we merely introduce the main extensions to the EPML to cater for the C-EPC specifications (see Table 1).


As can be seen from Table 1, for each configurable node we introduce an EPML representation element. A configurable function is defined as an extension to a regular EPC function in EPML, merely annotating a new attribute element `configuration`, which is optional and may take a value of `on`, `off`, or `opt`. Configurable connectors are likewise specified as extensions to regular connectors, with the option of setting the attribute element `configuration` to a concrete value – in accor-

dance to the definitions outlined in Section 2.2. Specifically, if for a configurable connector the value `seq` is selected, an attribute `goto` specifies the ID of an EPC node of the process model sequence selected. Configuration requirements and guidelines, respectively, are defined as logical expressions involving a number of configurable nodes. In EPML they are thus defined as part of the root `epc` element, with a list containing the IDs of involved elements (`idRefs`). The logical expressions themselves can be modeled via XPath expressions, *e.g.*

```
<configurationRequirement idRefs="2 4">
  <if xpath="function[@id='2']//configuration[@value='off']">
    <then xpath="function[@id='4']//configuration[@value='on']">
  </configurationRequirement>
```

Note that this specification allows for a representation of C-EPCs both *before* configuration (such as the one depicted in the left part of Fig. 1), and *after* configuration (such as the one depicted in the middle part of Fig. 1). Also, as our definitions are mere extensions to the traditional EPC specification in EPML, on the one hand traditional EPC models represented in EPML can also be validated against the extended EPML schema, and on the other hand EPML tools that are not aware of configuration aspects are still able to process C-EPCs as traditional EPCs by simply ignoring the additional configuration element information.

**Table 1.** EPML representations for C-EPC notation

C-EPC specification element	EPML representation
	<pre>&lt;xs:element name="configurableFunction"&gt;   &lt;xs:complexType&gt;     &lt;xs:choice minOccurs="0"&gt;       &lt;xs:element name="configuration"&gt;         &lt;xs:complexType&gt;           &lt;xs:attribute name="value" use="optional"&gt;             &lt;xs:simpleType&gt;               &lt;xs:restriction base="xs:string"&gt;                 &lt;xs:enumeration value="on"/&gt;                 &lt;xs:enumeration value="off"/&gt;                 &lt;xs:enumeration value="opt"/&gt;               &lt;/xs:restriction&gt;             &lt;/xs:simpleType&gt;           &lt;/xs:attribute&gt;         &lt;/xs:complexType&gt;       &lt;/xs:element&gt;     &lt;/xs:choice&gt;   &lt;/xs:complexType&gt; &lt;/xs:element&gt;</pre>

---

```
<xs:element name="configurableConnector" type="typeCOR">
  <xs:complexType>
    <xs:choice minOccurs="0">
      <xs:element name="configuration">
        <xs:complexType>
          <xs:attribute name="value" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="or"/>
                <xs:enumeration value="and"/>
                <xs:enumeration value="xor"/>
                <xs:enumeration value="seq"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="goto" type="xs:integer"/>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Ⓟ

---

```
<xs:element name="configurableConnector" type="typeCXOR">
  <xs:complexType>
    <xs:choice minOccurs="0">
      <xs:element name="configuration">
        <xs:complexType>
          <xs:attribute name="value" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="xor"/>
                <xs:enumeration value="seq"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="goto" type="xs:integer"/>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Ⓧ

---

```
<xs:element name="configurableConnector" type="typeCAnd">
  <xs:complexType>
    <xs:choice minOccurs="0">
      <xs:element name="configuration">
        <xs:complexType>
          <xs:attribute name="value" default="and"
            use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="and"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

---

ⓗ

---

```

<xs:element name="configurationRequirement"
  <xs:complexType>
    <xs:sequence>
      <xs:element name="if">
        <xs:complexType>
          <xs:attribute name="xpath" type="xs:string"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="then" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="xpath" type="xs:string"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="idRefs">
      <xs:simpleType>
        <xs:list itemType="xs:integer"/>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

```

---

```

<xs:element name="configurationGuideline">
  <xs:complexType>
    </xs:sequence>
      <xs:element name="if">
        <xs:complexType>
          <xs:attribute name="xpath" type="xs:string"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="then" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="xpath" type="xs:string"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="idRefs">
      <xs:simpleType>
        <xs:list itemType="xs:integer"/>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

```

---



Now, based on these EPML specifications, reference model configuration tool support may be designed that facilitates the model-driven configuration and translation of C-EPCs. In particular, the EPML specifications will be used to (a) leverage the modeling of C-EPCs via existing modeling tools, such as ARIS or the open source platform EPC Tools, (b) design a XML schema-based tool for checking the validity of configurations, (c) implement an EPML-based program for translating C-EPCs in EPCs, and (d) facilitate the interchange of configured (C-) EPCs to other process specification languages.

## 4 Summary & Conclusions

This paper reported on syntactical and semantic complications of reference model configuration, using the example of translating C-EPC models to lawful regular EPC models. We argued that configuration occurs before the background of the local environment of the configurable nodes in C-EPC models and showed that both a syntactical and semantic perspective must be considered when mapping configurable nodes

to desired regular EPC nodes. Resulting from these elaborations, we presented our initial conceptual work towards adequate tool support for the configuration of process models. Based on our research, adequate tool support can be designed that embeds our recommendations and thereby guides users when configuring Enterprise Systems based on configurable reference process models.

Our research has a few limitations. First, our conceptual approach needs to be empirically validated to prove its feasibility and applicability. We already conducted an initial laboratory experiment with postgraduate IT students at an Australian university on the perceived usefulness and perceived ease of use of C-EPCs in comparison to EPCs resulting in the finding that C-EPCs are in fact perceived as more useful and easier to use for the task of reference model configuration.<sup>2</sup> However, we still need to empirically test our work with real business practitioners. This task is currently underway. Second, we focused on the EPC notation and neglected the question of its executability. However, we selected the EPML interchange format as a basis for our conceptual design of tool support for good reason, as it denotes an interchange format for various process modeling languages and may hence facilitate such translation from (C-) EPC models to executable process specifications.

## Acknowledgements

We appreciate the continuous fruitful contributions of Alexander Dreiling and Wasim Sadiq to the C-EPC research project and of Markus Nüttgens to the EPML initiative.

## References

1. Scott, J.E., Vessey, I.: Managing risks in enterprise systems implementations. *Communications of the ACM* 45 (2002) 74-81
2. Sabherwal, R., Chan, Y.E.: Alignment Between Business and IS Strategies: A Study of Prospectors, Analyzers, and Defenders. *Information Systems Research* 12 (2001) 11-33
3. Davenport, T.H., Short, J.E.: The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review* 31 (1990) 11-27
4. Hammer, M., Champy, J.: *Reengineering the Corporation: A Manifesto for Business Revolution*. Harpercollins, New York (1993)
5. Keller, G., Nüttgens, M., Scheer, A.-W.: Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". *Arbeitsberichte des Instituts für Wirtschaftsinformatik* 89. Institut für Wirtschaftsinformatik der Universität Saarbrücken, Saarbrücken (1992)
6. Rosemann, M.: Using reference models within the enterprise resource planning lifecycle. *Australian Accounting Review* 10 (2000) 19-30
7. Curran, T., Keller, G., Ladd, A.: *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Prentice Hall PTR, Upper Saddle River (1997)
8. Rosemann, M., van der Aalst, W.: A Configurable Reference Modelling Language. *Information Systems to appear* (2005)

---

<sup>2</sup> The design and outcomes of the laboratory experiment are available from the authors on request.

9. Mendling, J., 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, Vienna (2005)
10. van der Aalst, W., Desel, J., Kindler, E.: On the semantics of EPCs: A vicious circle. In: Nüttgens, M., Rump, F.J. (ed.): Proceedings of the GI-Workshop und Arbeitskreistreffen EPK 2002 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Trier (2002) 71-79
11. van der Aalst, W.: Formalization and Verification of Event-driven Process Chains. *Information and Software Technology* 41 (1999) 639-650
12. Nüttgens, M., Rump, F.J.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: Desel, J., Weske, M. (ed.): Proceedings of the GI-Workshop und Fachgruppentreffen Promise 2002 - Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen. *Lecture Notes in Informatics*, Vol. P-21. Gesellschaft fuer Informatik, Potsdam (2002) 64-77
13. Langner, P., Schneider, C., Wehler, J.: Petri Net Based Certification of Event-Driven Process Chains. In: Desel, J., Silva, M. (ed.): Proceedings of the 19th International Conference on Application and Theory of Petri Nets. *Lecture Notes in Computer Science*, Vol. 1420. Springer, Lisbon (1998) 286-305
14. Dehnert, J., Rittgen, P.: Relaxed Soundness of Business Processes. In: Dittrich, K.R., Gerpert, A., Norrie, M.C. (ed.): Proceedings of the 13th International Conference on Advanced Information Systems Engineering. *Lecture Notes In Computer Science*, Vol. 2068. Springer, Interlaken (2001) 151-170
15. Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuroпка, D.: Configurative Process Modeling - Outlining an Approach to increased Business Process Model Usability. In: Khosrow-Pour, M. (ed.): Proceedings of the 14th Information Resources Management Association International Conference. IRM Press, New Orleans (2004) 615-619
16. Soffer, P., Golany, B., Dori, D.: ERP modeling: a comprehensive approach. *Information Systems* 28 (2003) 673-690
17. Gulla, J.A., Brasethvik, T.: On the Challenges of Business Modeling in Large-Scale Reengineering Projects. In: Chen, P.P., Embley, D.W., Kouloumdjian, J., Liddle, S.W., Roddick, J.F. (ed.): Proceedings of the 4th International Conference on Requirements Engineering. IEEE, Schaumburg (2000) 17-26
18. Dreiling, A., Rosemann, M., van der Aalst, W., Sadiq, W., Khan, S.: Model-Driven Process Configuration of Enterprise Systems. In: Sinz, E.J., Ferstl, O.K. (ed.): Proceedings of the 7th International Tagung Wirtschaftsinformatik. Gesellschaft für Informatik, Bamberg (2005) 691-710
19. Mendling, J., Nüttgens, M.: EPC Syntax Validation with XML Schema Languages. In: Nüttgens, M., Rump, F.J. (ed.): Proceedings of the 2nd GI Workshop on Event-Driven Process Chains. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Bamberg (2003) 19-30
20. Mendling, J., Nüttgens, M.: Exchanging EPC Business Process Models with EPML. In: Nüttgens, M., Mendling, J. (ed.): Proceedings of the 1st GI Workshop XML4BPM - XML Interchange Formats for Business Process Management. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Marburg (2004) 61-79
21. Mendling, J., Nüttgens, M.: Transformation of ARIS Markup Language to EPML. In: Nüttgens, M., Rump, F.J. (ed.): Proceedings of the 3rd GI Workshop on Event-Driven Process Chains. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Luxembourg (2004) 27-38