Exchanging EPC Business Process Models with EPML

Jan Mendling

Abteilung für Wirtschaftsinformatik
Wirtschaftsuniversität Wien,
A-1090 Wien
jan.mendling@wu-wien.ac.at

Markus Nüttgens

Universität des Saarlandes
D-66041 Saarbrücken
markus@nuettgens.de

Abstract: In this paper EPML is presented as an interchange format for EPC business process models. EPML builds on EPC syntax related work and is designed to be applicable as a serialisation format for EPC modelling tools. After a description of EPML in the large, examples are given to illustrate selected representational aspects including flat and hierarchical EPCs, business views, and graphical information.

1.   Exchanging Business Process Models

Today business process modelling is mainly used in two different contexts: business analysts use process models for documentation purposes, for process optimization and simulation; while information system analysts use them on the middleware tier in order to glue together heterogeneous systems. For both of these layers analysts have a variety of tools to choose from in order to support modelling of processes. In 2002 Gartner Research distinguishes 35 major vendors of such software [Ga02]. Heterogeneity of these tools causes huge interoperability problem in this context. A recent survey of DelphiGroup [De03] identifies the lack of a common and accepted interchange format for business process models as the major detriment for business process management.

Event-Driven Process Chains (EPC) [KNS92] are a wide-spread method for business process modelling. SAP AG has been using them to express their SAP reference model [Ke99]. Motivated by the heterogeneity of business process modelling tools, a proposal for an interchange format for EPCs is in progress of development. It is called EPC Markup Language (EPML) [MN02, MN03b, MN03c]. The establishment of a standardized representation of business process models may be even more beneficial than in other domains of standardization, because it may be used in two different directions: horizontal interchange will simplify the integration of BPM tools of the same scope. Vertical interchange can leverage the integration of simulation engines, execution engines, and monitoring engines [Wf02]. Standardization might be a crucial step to close the engineering gap between business process modelling and implementation.

This paper gives an overview over EPML. Section 2 introduces Event-driven Process Chains as a method to express business process models, their syntactical elements, and related research on EPC syntax. Section 3 presents EPML general design principles and XML design guidelines that have guided the specification. Section 4 explains how the syntax elements of EPML relate to each other and outlines why edge element lists are used to describe EPC process graphs in EPML. The Sections 5 to 8 introduce specific aspects of EPML by giving examples: Section 5 presents a simple EPC example and its EPML syntax representation; Section 6 shows how hierarchies of EPCs are expressed; Section 7 discusses how business perspectives can be included in an EPML file; and Section 8 shows which graphical information can be attached to a process element Section 9 concludes and lists future directions of research.

2. Event-Driven Process Chains (EPCs)

Most of the formal contributions on EPCs have been focused on semantics, especially on the semantics of OR connectors. The translation of EPC process models to Petri Nets plays an important role in this context. Examples of this research can be found in Chen/Scheer [CS94], Langner/Schneider/Wehler [LSW98], van der Aalst [Aa99], Rittgen [Ri00], and Dehnert [De02]. A major point of discussion is the "non-locality" of join-connectors [ADK02]. This aspect has recently been formalized by Kindler [Ki03]. In this paper we will focus on EPC syntax referencing to based on the syntax definition of EPCs in [NR02]. Therefore we give a brief survey of syntax related work.

In Keller/Nüttgens/Scheer the EPC is introduced [KNS92] to represent temporal and logical dependencies in business processes. Elements of EPCs may be of function type (active elements), event type (passive elements), or of one of the three connector types AND, OR, or XOR. These objects are linked via control flow arcs. Connectors may be split or join operators, starting either with function(s) or event(s). These four combinations are discussed for the three connectors resulting in twelve possibilities. OR-Split and XOR-Split are prohibited after events, due to the latter being unable to decide which following functions to choose. Based on practical experience with the SAP Reference model, process interfaces and hierarchical functions are introduced as additional element types of EPCs [KM94]. These two elements permit to link different EPC models: process interfaces can be used to refer from the end of a process to a following process, hierarchical functions allow to define macro-processes with the help of sub-processes. Keller [Ke99] and Rump [Ru99] provide a formal approach defining the EPC syntax. Based on this, Nüttgens/Rump [NR02] distinguish the concepts of a flat EPC Schema and a hierarchical EPC Schema. A flat EPC Schema is defined as a directed and coherent graph with cardinality and type constraints. A hierarchical EPC Schema is a set of flat or hierarchical EPC Schemata. Hierarchical EPC Schemata consist of flat EPC Schemata and a hierarchy relation linking either a function or a process interface to another EPC Schema. Fig. 1 shows a hierarchical EPC Schema consisting of two processes, which are linked via a hierarchical relation attached to the process interface "To Design Process".
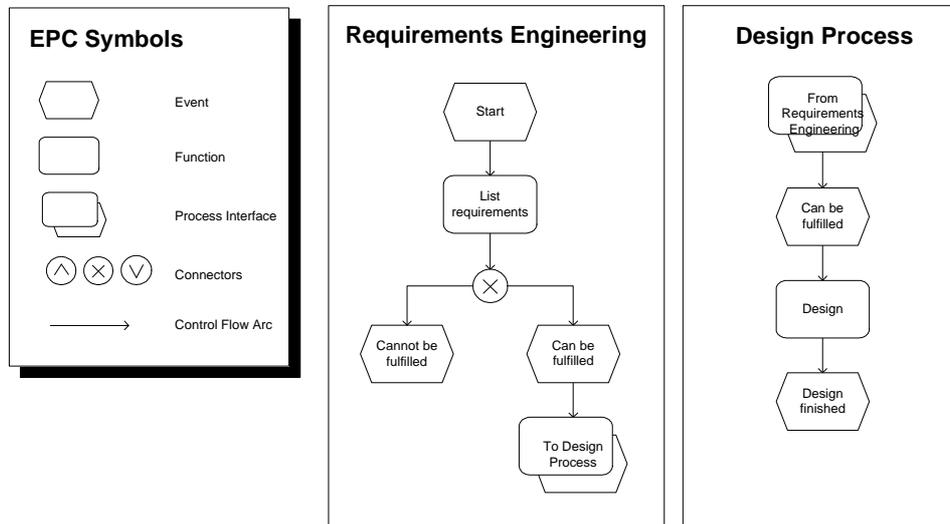
Fig. 1. EPC example of a simple requirements engineering process. The connector represents an "exclusive or". After "Can be fulfilled" a process interface links to the design process.

Work on EPML started off in 2002 mainly inspired by heterogeneity of business process modelling tools and the potential of efficiency gains for the use of an intermediary format [WHB02, MN03b]. As a first step, comparable efforts towards standardized interchange formats in the area of Petri Nets, BPML, and UML have been analyzed [MN02]. Work on syntactical correctness led to a revised EPC syntax definition based on implicit arc types and related syntax properties [MN03a]. An analysis on EPC syntax validation discusses in how far these syntax properties can be expressed via standard XML Schema languages [MN03c]. A proposal for an EPML schema is presented in [MN04] and has been made available at http://wi.wu-wien.ac.at/~mendling/EPML.

3. Design Principles

Towards the definition of an XML syntax for EPC models, the global goal of defining a tool and platform independent XML-based interchange format for EPCs has to be translated into domain-specific design principles in order to derive design decisions. Domain-independent XML design guidelines standardize the way how things are put into XML syntax.

3.1. EPML General Design Principles

In order to put EPML design principles into context, we present design principles proposed for ASC X12 Reference Model for XML Design (X12) [AN02] and Petri Net Markup Language (PNML) [Bi03]. X12 is a specification describing a seven layer model for the development of business documents. The definition of X12 was guided by

four high level design principles: alignment with other standards, simplicity, prescriptiveness, and limit randomness. *Alignment* with other standards refers to the specific domain of business documents where other organisations including OASIS and UN/CEFACT, World Wide Web Consortium, and OASIS UBL also develop specifications. *Simplicity* is a domain independent principle. It demands features and choices to be reduced to a reasonable minimum. *Prescriptiveness* is again related to business documents. This principle recommends one to define rather more precise and specific business documents than too few which are very general. *Limit randomness* addresses certain constructs in XML schema languages that provide multiple options and choices. These aspects shall be limited to a minimum. The PNML approach for Petri Nets is governed by the principles flexibility, no ambiguity, and compatibility [Bi03]. *Flexibility* is an important aspect for Petri Nets, because all kinds of currently discussed and also prospective classes of Petri Nets shall be stored. This will be achieved with labels which can be attached to arcs and nodes. *No ambiguity* refers to the problem of standardized labels. Therefore, Petri Net Type Definitions define legal labels for particular net types. *Compatibility* deals with the problem of semantically equivalent labels used by different Petri net types. These overlapping labels shall be exchangeable.

The EPML approach reflects these different design principles. It is governed by the principles of readability, extensibility, tool orientation, and syntactical correctness [MN03b]. *Readability* expects EPML elements and attributes to have intuitive and telling names. This is important because EPML documents will be used not only by applications, but also by humans who write XSLT-scripts that transform between EPML and other XML vocabularies. Readability is partially related to simplicity and limited randomness of the X12 approach. *Extensibility* reflects a problem that is analogous to different types of Petri nets. An important aspect of BPM is to provide different business perspectives and views on a process. EPML should be capable to express arbitrary perspectives instead of only supporting a pre-defined set. Section 6 is dedicated to this issue. *Tool orientation* deals with graphical representation of EPCs. This is a crucial feature, because BPM tools provide a GUI for developing models. EPML should be able to store various layout and position information for EPC elements. Finally, *syntactical correctness* summarizes aspects dealing with EPC syntax elements and their interrelation. The following paragraph will discuss general XML design aspects.

3.2. XML Design Guidelines

Basically, two general approaches towards XML design guidelines can be distinguished: a theoretical one building on normal forms and information content measures like entropy; and a pragmatic one giving advise on when to use which XML language concepts and how to name elements and attributes.

The *theoretical* approach builds on insights from database theory. For relational database models concepts like functional dependency (FD), multi-value dependency (MVD), and join dependency (JD) have been formally described [Bi95]. In order to derive schemas with good properties, decomposition algorithms have been developed to achieve different levels of normal forms. These normal forms avoid redundancies and anomalies

from operations on relational data. Analogously, a normal form has been presented for XML, called (XNF) [EM01, AL02]. In [AL03] an information-theoretic approach is presented that bridges the conceptual gap between relational and XML representations. A theory is developed building on entropy measures that brings forth a concept-independent understanding of the interrelation of redundancies and normal forms. A schema is called *well-designed* when it cannot contain instance data with an element that has less than maximum information in terms of conditional entropy [AL03]. From this it can be shown that a schema which has only FDs and neither MVDs nor JD is well-designed iff (if and only if) it is in Boyce-Codd-Normal Form. FD for XML schemas occur when paths from the root to nodes in the XML tree depend upon other paths. Analogously, an XML schema subject to FDs is well-designed iff it is in XNF [AL03]. A violation of XNF implies redundancies in that sense that a path may reach different nodes, but that these nodes all have the same value. Such violations can be cured by a normalization algorithm that moves attributes and creates new elements until XNF is achieved [AL03]. For XML reference model design this implies that there should be no XPath [CD99] statement that always returns a set of nodes all containing the same value. Then the XNF condition is fulfilled and the schema is well-designed.

Pragmatic approaches deal with extensibility and design leeway in XML. Documents from ISO [ISO01], SWIFT [SW01], and X12 [AN02] establish design rules in order to minimize ambiguity and maximize communicability of XML schemas. Pragmatic XML design guidelines include conventions for names; for the choice of style between elements and attributes; for the use of special schema language features; and for namespace support. *Naming conventions* refer to the choice of element and attribute names. ISO, SWIFT, MISMO, and X12 agree on using English words for names. Names may also consist of multiple words in so-called Upper Camel Case (no separating space, each new word beginning with a capital letter) according to MISMO, SWIFT, and ISO, abbreviations and acronyms shall be limited to a minimum. *Style conventions* govern the choice between elements and attributes. X12 recommends the usage of attributes for metadata and elements for application data [AN02]. In this context, it is a good choice to understand identifying keys as metadata and put them into attributes. That allows a DTD conforming usage of the ID, IDREF, and IDREFS data types and a respective key or keyref declaration in a W3C XML Schema [Be01, BM01]. Further, attributes are considered to provide a better readability of content [AN02]. Therefore, content that can never be extended may also be put into attributes. *Schema conventions* recommend one to use only a reduced set of the expressive power provided by an XML schema language. X12 advises one to avoid mixed content, substitution groups, and group redefinition from another schema; one should use only named content types and built-in simple types, to name but a few aspects. We refer to [AN02] for a broader discussion. *Namespace conventions* refer to the usage of namespaces in instance documents. X12 recommends one to use explicit namespace references only at the root level. Theoretical and pragmatic approaches offer complementary guidelines for the development of "good" XML schemas. The guidelines presented have contributed to the EPML proposal.

4. EPML in the Large

`<epml>` is the root element of an EPML file. Like all other elements it may have `<documentation>` or `<toolInfo>` child elements. These may contain data that has been added by the editor of the EPML file or tool specific data attached by an application. These two elements are of XML Schema type anyType which means that they may hold arbitrary nesting of XML data. It is recommended to use only standardised Dublin Core Metadata Elements [DC03] for documentation of the EPML file, and to add only such application specific data that has relevance for the internal storage of models in a certain tool, but which does no influence the graphical presentation of a model. General graphic settings may be defined in the `<graphicsDefault>` element. The `<coordinates>` element is meant to explicate the interpretation of coordinates annotated to graphical elements of an EPC. The `@xOrigin` attribute may take the values "leftToRight" or "rightToLeft", and the `@yOrigin` attribute can hold "topToBottom" or "bottomToTop". It is recommended to always use the "leftToRight" and "topToBottom" settings which most of the tools assume. Yet, there are still exceptions like MS Visio [Mi03] that has its y-axis running from the bottom of the screen upward. It is recommended to transform these coordinates when storing EPC models in EPML.

In [NR02] an EPC Schema Set is defined as a set of hierarchical EPC Schemas. Each of these hierarchical EPC Schemas consists of a flat EPC Schema which may have hierarchy relations attached with functions or process interfaces. The detailed discussion of flat EPC Schemas is left to the following Section; here, it is sufficient to have a general understanding of what EPCs are. Syntactically, a hierarchy relation connects functions or process interfaces with other EPC processes. Semantically, it refers to the call of sub-processes. `<epml>` also has a `<definitions>` child element which is explained in conjunction with the `<directory>` element. The `<view>` element allows business views and perspectives to be defined. Its `<unit>` element is a container for information about an entity which is important in a business process. This unit may be attached to control flow elements. Figure 2 gives an overview over EPML via a metamodel; Table 1 illustrates the content model of high-level EPML elements.

In EPML a hierarchy of processes is organised by the help of directories. A `<directory>` holds a `@name` attribute, other directories, and/or EPC models. Each `<epc>` is identified by an `@epcId` attribute and has a `@name` attribute. The `@epcId` can be referenced by hierarchy relations attached to functions or process interfaces. The EPC control flow elements will be discussed in paragraph 4.2. In a hierarchy of EPC models there may be the problem of redundancy. An EPC process element might be used in two or more EPC models. In such a case there should be a place to store it once and reference it from the different models. This is precisely the aim of the `<definitions>` element. It serves as a container for control flow elements that are used more than once in the model hierarchy.
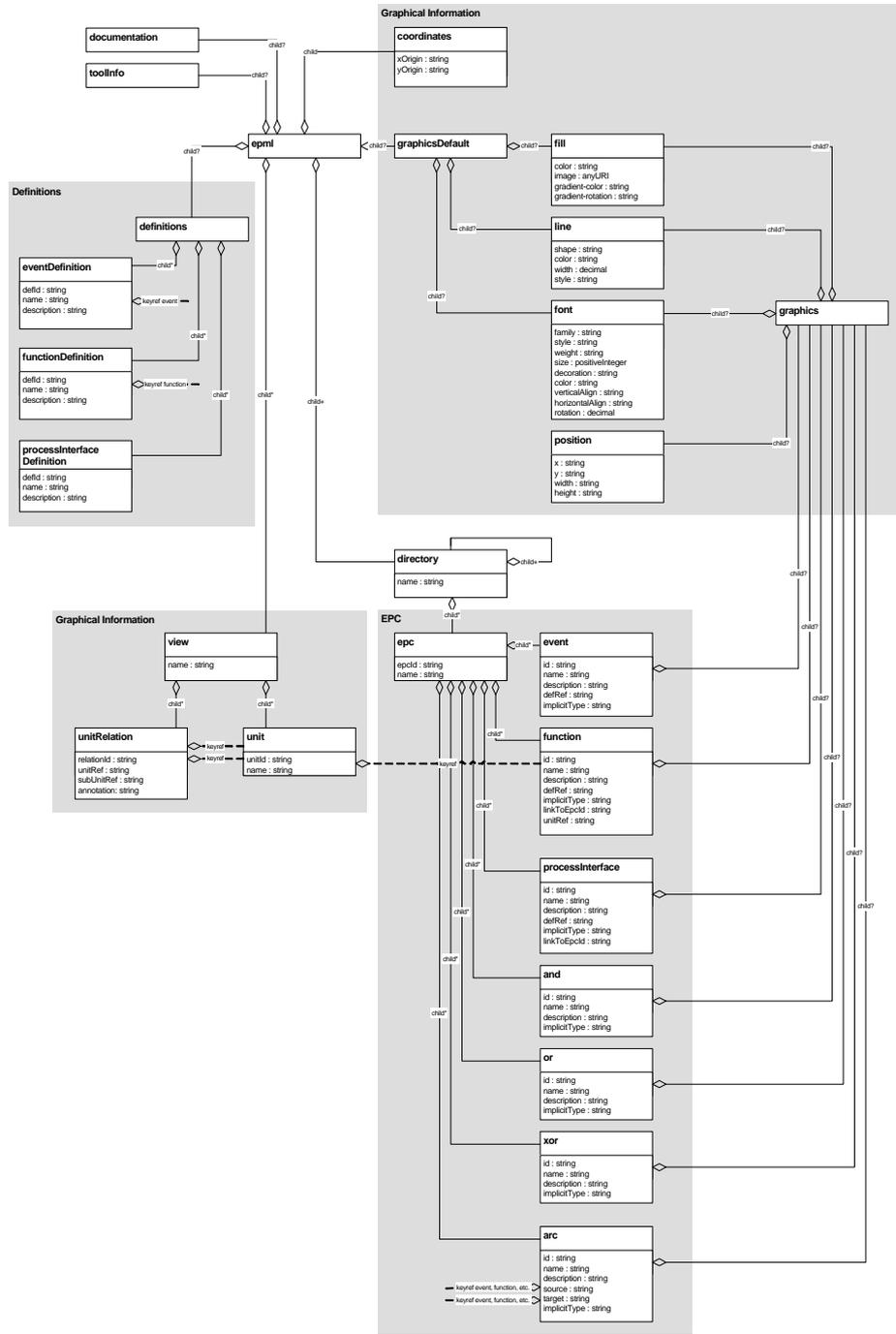
Graphical Information

documentation

child?

toolInfo

child?

coordinates

xOrigin : string
yOrigin : string

child2

epml

child? child? child?

child?

graphicsDefault

child?

fill

color : string
image : anyURI
gradient-color : string
gradient-rotation : string

child?

Definitions

definitions

child*

eventDefinition

defId : string
name : string
description : string

keyref event

child*

functionDefinition

defId : string
name : string
description : string

keyref function

child*

**processInterface
Definition**

defId : string
name : string
description : string

child*

child? **line**

shape : string
color : string
width : decimal
style : string

child?

child? **font**

family : string
style : string
weight : string
size : positiveInteger
decoration : string
color : string
verticalAlign : string
horizontalAlign : string
rotation : decimal

child?

graphics

child?

position

x : string
y : string
width : string
height : string

child* child+

directory

name : string

child+

child*

Graphical Information

view

name : string

child? child*

unitRelation

relationId : string
unitRef : string
subUnitRef : string
annotation: string

keyref
keyref

unit

unitId : string
name : string

keyref

EPC

epc

epcId : string
name : string

child* **event**

id : string
name : string
description : string
defRef : string
implicitType : string

child?

child* **function**

id : string
name : string
description : string
defRef : string
implicitType : string
linkToEpcId : string
unitRef : string

child?

child* **processInterface**

id : string
name : string
description : string
defRef : string
implicitType : string
linkToEpcId : string

child?

child* **and**

id : string
name : string
description : string
implicitType : string

child?

child* **or**

id : string
name : string
description : string
implicitType : string

child?

child* **xor**

id : string
name : string
description : string
implicitType : string

child?

child* **arc**

id : string
name : string
description : string
source : string
target : string
implicitType : string

child?

keyref event, function, etc.
keyref event, function, etc.

Fig. 2: Overview over EPML including its main syntax elements.

EPML element	Attributes and Sub-Elements

`<epml>`	`<documentation> ?`
	`<toolInfo> ?`
	`<graphicsDefault> ?`
	`<coordinates>`
	`<definitions>`
	`<view> *`
	`<directory> +`

`<definitions>`	`<documentation> ?`
	`<toolInfo> ?`
	`<eventDefinition> *`
	`<functionDefinition> *`
	`<processInterfaceDefinition> *`

`<directory>`	`@name`
	`<documentation> ?`
	`<toolInfo> ?`
	`<directory> *`
	`<epc> *`

`<epc>`	`@epcId, @name`
	`<documentation> ?`
	`<toolInfo> ?`
	`<event> *`
	`<function> *`
	`<processInterface> *`
	`<and>, <or>, <xor> *`
	`<arc>`

Table 1: High level elements of an EPML file.

5. Flat EPCs in EPML

This Section describes how a simple flat EPC process is encoded in EPML. Figure 3 shows the example of an "Online Shopping" process. After starting the process, a product is added to the shopping cart. When the buyer wants to buy more, he adds another product to the shopping cart until the list of products is completed. He then completes the order by stating her shipping address. The code on the right hand side of Fig. 3 shows an excerpt from an EPML file corresponding to that process. The root tag of every EPML file is `<epml>` and it must belong to the EPML namespace. The directory tag contains one EPC model which has the name "Online Shopping" and the

ID "1". The EPC tag serves as a container of an unordered set of EPC control flow elements. All of the latter have a unique ID attribute. The name tag of the events and functions carry the text which is displayed as the label of the respective symbol in the process diagram. Arcs are modelled as individual elements with source and target attributes. This way of process graph representation is called edge element list [MN04]. It is also used by Graph eXchange Language (GXL) [WKR02]; by Petri Net Markup Language (PNML) [WK02]; by MS Visio's XML-based VDX format [Mi03]; XML Metadata Interchange for UML models [OMG03]; and XML Process Definition Language (XPDL) from Workflow Management Coalition (WfMC) [Wf02]. In contrast the Business Process Modeling Language (BPML) [Ar02] and the Business Process Execution Language for Web Services (BPEL4WS) [An03] use a block-oriented representation. AML, the XML format of ARIS Toolset [IDS01] uses adjacency sub-element lists which are attached to the source node of an arc. Arcs and connectors are not required to have a name. A complete list of EPC control flow elements and their sub-elements is presented in Table 2. The following Section illustrates the representation of hierarchical EPC Schemas in EPML.


Fig. 3: A simple Online Shopping Process and parts of its EPML representation.

EPML element	Attributes and Sub-Elements

`<event>`	`@id`
	`<name>`
	`<description>`
	`<reference @defRef> ?`
	`<graphics> ?`
	`<syntaxInfo @implicitType> ?`

`<function>`	`@id`
	`<name>`
	`<description>`
	`<reference @defRef> ?`
	`<graphics> ?`
	`<syntaxInfo @implicitType> ?`
	`<toProcess @linkToEpcId> ?`
	`<unitReference @unitRef @role> ?`

`<processInterface>`	`@id`
	`<name>`
	`<description>`
	`<reference @defRef> ?`
	`<graphics> ?`
	`<syntaxInfo @implicitType> ?`
	`<toProcess @linkToEpcId> ?`

`<and>`, `<or>`, `<xor>`	`@id`
	`<name> ?`
	`<description> ?`
	`<graphics> ?`
	`<syntaxInfo @implicitType> ?`

`<arc>`	`@id`
	`<name> ?`
	`<description> ?`
	`<flow @source @target> ?`
	`<graphics> ?`
	`<syntaxInfo @implicitType> ?`

Table 2: Control flow elements of an EPML file.

6. Hierarchical EPCs in EPML

Consider an extension of the example above. After the "Online Shopping" process has been modelled, the function "Add Product to Shopping Cart" is refined by a sub-process called "Product Selection". This means that the EPML file has to include this new process and the hierarchical relation between the function and the sub-process. Figure 4 illustrates the EPML representation of EPC processes with hierarchy relations. The code includes an excerpt from the "Online Shopping" process described in Fig. 3. The hierarchy relation is described via sub-element of the function tag which is called `<toProcess>`. This element has a `@linkToEpcId` pointing to the "Product Selection" process which has an `@epcId` of 2. Hierarchy relations of process interfaces are also described by a `<toProcess>` element. In that situation the process interface at the end of a process points to a start-process interface of another process. The epcId attribute of a process is unique for the whole EPML file. EPC models may be organized in a hierarchy of directories. Hierarchy relations are allowed between processes no matter where they are placed in the directory hierarchy, as long as hierarchy relations are acyclic. In order to avoid redundancies, multiple occurrences of a function, an event, or a process interface can be defined in the `<definitions>` block. I.e. a function used twice in a process hierarchy should contain a `<reference>` sub-element pointing to a function definition that stores its parameters in the definitions block.
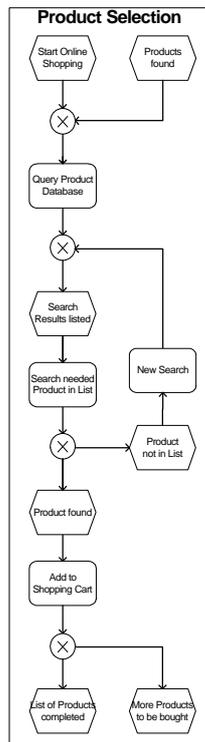

```xml
<?xml version="1.0" encoding="UTF-8"?>
<epml:epml xmlns:epml="http://www.epml.de">
    ...
    <directory name="Root">
        <epc epcId="1"
            name="Online Shopping">
            ...
            <function id="3">
                <name>Add Product to
                Shopping Cart</name>
                <toProcess linkToEpcId="2"/>
            </function>
            <arc id="12">
                <flow source="3" target="4"/>
            </arc>
            <or id="4"/>
            ...
        </epc>
        <epc epcId="2"
            name="Product Selection">
            ...
        </epc>
    </directory>
</epml:epml>
```
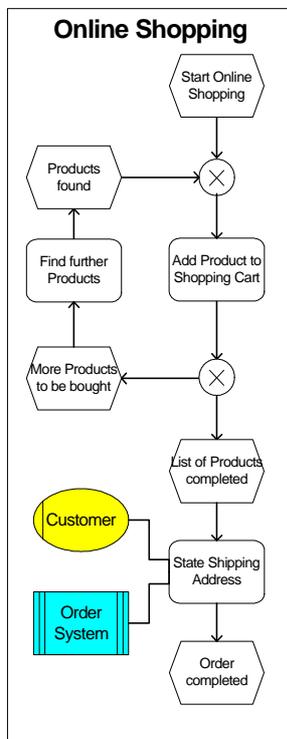
Fig. 4: The Product Selection process – a sub-process of the Online Shopping Process.

7. Business Perspectives and Views

Business perspectives and views play an important role for the analysis and conception of process models, especially for EPCs. Perspectives have proven valuable to partition the specification of a complex system [Fi92]. There have been many different perspectives proposed for business process modelling. The Architecture of Integrated Systems (ARIS) extends the EPC with a data-oriented, a functional, an organisational, an application-oriented, and a product/service-oriented perspective [Sc00]. The PROMET concept differentiates between business dimensions explicitly including organisation, data, functions, and personnel [Ös95]. An in-depth survey of organisational entities provided in workflow management systems is given in [RM98]. The link between role-based access control (RBAC) and business scenarios is analysed in [NS02] and a methodology to generate role hierarchies is developed. From a delegation perspective [AKV03] structure the organisational perspective of a workflow system into a meta model including resources, organisational units, users, and roles. In [Wh03] and [BAN03] swim lanes and pools are recommended as a metaphor for the graphical representation of parties involved in a process. Recently, BPM languages like BPEL4WS contain references to WSDL descriptions [Ch01] of Web Services as a new category of resource perspectives. Beyond resources there have been further perspectives proposed like e.g. risk [BO02], performance measurement [IDS03] to name but a few. The OWL-S Initiative strives to develop a standardised business process ontology for Web Service [OW04]. This is a difficult task taken into consideration the variety of possible perspectives and views. There are even doubts whether a standardised ontology is desirable, because different domains and different business sectors need tailor-made meta models that best fit their specific business model [KK02].

EPML element	Attributes and Sub-Elements

`<view>`	`@name`
	`<unit> *`
	`<unitRelation> *`

`<unit>`	`@unitId` `@name`

`<unitRelation>`	`@relationId`
	`@unitRef`
	`@subUnitRef`
	`@annotation ?`

`<unitReference>`	`@unitRef`
	`@role ?`
	`@value ?`

Table 3: Business perspectives and views in EPML.

These arguments have governed the decision of letting EPML be guided by the principle of extensibility instead of standardising certain views. The `<view>` element is meant to be a container of entities of a certain business perspective and their relationships (cf. Table 3). The `<unit>` element describes an entity within the domain of a business view by a `@unitId` and a `@name`. The `<unitRelation>` expresses a hierarchical relationship between by the help of a `@unitRef` and a `@subUnitRef`. The `@annotation` may be used to detail the kind of relationship between the units. There is also a `@relationId` included in order to logically distinguish different relationships between two of the same units. Function elements of a control flow may contain a `<unitReference>`. The `@role` and the `@value` attribute allow one to specify additional information concerning the relationship between the function and the unit.

```
<?xml version="1.0" encoding="UTF-8"?>
<epml:epml xmlns:epml="http://www.epml.de">
    ...
    <view name="Party>
        <unit unitId="u1" name="Customer"/>
    </view>
    <view name="System">
        <unit unitId="u2" name="Order System"/>
    </view>
    <directory name="Root">
        <epc epcId="1"
            name="Online Shopping">
            ...
            <function id="7">
                <name>State Shipping Address
                </name>
                <unitReference unitRef="u1"
                    role="performs activity"/>
                <unitReference unitRef="u2"
                    role="receives data record"/>
            </function>
            ...
        </epc>
    </directory>
</epml:epml>
```

Fig. 5: The Online Shopping process including a "Customer" and an "Order System" unit.

Figure 5 illustrates the use of view and unit elements in the "Online Shopping" process of our example. In the header of the EPML file the views "Party" and "System" are declared. Each of these views has one unit: "Customer" is a "Party", and "Order System" is a "System" involved in the process. Within the function element these units are referenced via a `<unitReference>` element. The first one describes that the unit "u1" (the customer) takes the role "performs activity" for the function "State Shipping Address". The second illustrates that the "Order System" receives a data record from this function. This mechanism can be used to declare arbitrary views for an EPC process.

8.   Graphical Information

Graphical Information refers to the presentation of EPC models in graphical BPM tools. This is a topic that is not special to EPML. The Petri Net Markup Language (PNML) has worked out and included a proposal for graphical information to be exchanged between modelling tools [Bi03]. This concept is also well suited for EPML and adopted here. There are some small modifications that will be made explicit in the discussion of the details. Similar to the `<graphics>` element of control flow objects, the top level element `<graphicsDefault>` may contain `<fill>`, `<line>`, and `<font>` default settings, but no `<position>` element.

EPML element	Attributes and Sub-Elements
`<graphics>`	`<position>` `<fill>` `<line>` `<font>`
`<position>`	`@x`, `@y`, `@width`, `@height`
`<fill>`	`@color`, `@image`, `@gradient-color`, `@gradient-rotation`
`<line>`	`@shape`, `@color`, `@width`, `@style`
`<font>`	`@family`, `@style`, `@weight`, `@size`, `@decoration`, `@color`, `@verticalAlign`, `@horizontalAlign`, `@rotation`

Table 4: The graphics element of an EPML file.

All the four attributes of the `<position>` element refer to the smallest rectangle parallel to the axes that can be drawn to contain the whole polygon symbolizing the object. The `@x` and `@y` attributes of the object describe the offset from the origin of the coordinates system of that angle of the object that is closest to the origin. The `@width` and the `@height` describe the length of the edges of the container rectangle. In PNML a separate dimension element is used to represent width and height. Arcs may have multiple position elements to describe anchor point where the arc runs through. Position elements of arcs do not have width and height attributes.

The <fill> element describes the appearance of the interior of an object. Arcs do not have fill elements. The @color attribute must take a RGB value or a predefined colour of Cascading Stylesheets 2 (CSS2) [Bo98]. In order to describe a continuous variation of the filling colour an optional @gradient-color may be defined. The @gradient-rotation sets the orientation of the gradient to vertical, horizontal, or diagonal. If there is the URI of an image assigned to @image the other attributes of fill are ignored. The <line> element defines the outline of an object. The @shape attribute refers to how arcs are displayed: the value "line" represents a linear connection of anchor points to form a polygon; the value "curve" describes a quadratic Bezier curve. The <font> element holds @family, @style, @weight, @size, and @decoration attributes in conformance with CSS2. In addition to PNML, there may be a font colour defined. @verticalAlign and @horizontalAlign specify the alignment of the text. In PNML the align attribute corresponds to the EPML horizontalAlign attribute, and verticalAlign is covered by a PNML offset element. @rotation describes a clockwise rotation of the text similar to the concept in PNML.

9. Outlook

Throughout this paper we have outlined how EPML can be used to store an exchange EPC business process models. Yet, there is still much discussion needed within the EPC community to achieve a consensus on EPC representation in EPML, and to leverage EPML application. There are several issues that will be addressed in the future. Firstly, in order to leverage the benefits of EPML as an interchange format, transformation scripts will be developed from major BPM tools towards EPML and reverse. A second issue is the graphical presentation. For PNML there already exists a transformation script to Scalable Vector Graphics (SVG) [Fe03]. A similar script will be developed from EPML to SVG. Thirdly, an XSLT-based [Cl99] syntax checker will be developed and continue the efforts of an XML-based syntax validation of EPCs [MN03c]. Finally, there is still much research needed to come to a general understanding of business perspectives for BPM. Methodologically, this will have to take meta modelling and semantic web techniques into account; furthermore related research on concrete perspectives will have to be consolidated. Administration of decentralized, loosely coupled models will be one of the topics in this context. In this sense, the development of EPML can – beyond its principle purpose as an interchange format – serve as a catalyst and a framework for the discussion of all these related topics. Information, on EPML can be found at http://wi.wu-wien.ac.at/~mendling/EPML/.

References

[Aa99]   van der Aalst, W.M.P.: Formalization and Verification of Event-driven Process Chains, in: Information and Software Technology 41(1999)10, pp. 639-650.

[ADK02] van der Aalst, W.; Desel, J.; Kindler, E.: On the semantics of EPCs: A vicious circle, in: Nüttgens, M.; Rump, F.J. (eds.): Geschäftsprozessmanagement mit Ereignisgesteuerten

Prozessketten - EPK 2002, Proceedings of the GI-Workshop EPK 2002, Trier, 2002, pp. 71-79.

[AKV03] van der Aalst, W.M.P.; Kumar, A.; Verbeek, H.M.W.: Organizational Modeling in UML and XML in the Context of Workflow Systems. In: Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), 2003, pp. 603-608.

[AL02] Arenas, M.; Libkin, L.: A normal form for XML documents. In: Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02), 2002, pp. 85-96.

[AL03] Arenas, M.; Libkin, L.: An Information-Theoretic Approach to Normal Forms for Relational and XML Data. In: Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'03), 2003, pp. 15-26.

[AN02] ANSI (ed.): ASC X12 Reference Model for XML Design, July 2002. http://www.x12.org/x12org/comments/X12Reference_Model_For_XML_Design.pdf.

[An03] Andrews, T.; Curbera, F.; Dholakia, H.; Goland, Y.; Klein, J.; Leymann, F.; Liu, K.; Roller, D.; Smith, D.; Thatte, S.; Trickovic, I.; Weerawarana, S.: Business Process Execution Language for Web Services (BPEL4WS) Version 1.1. BEA, IBM, Microsoft, SAP, Siebel, 2003.

[Ar02] Arkin, A.: Business Process Modeling Language (BPML). BPMI.org, 2002.

[BAN03] Becker, J.; Algermissen, L.; Niehaves, B.: Prozessmodellierung in eGovernment-Projekten mit der eEPK. In: Nüttgens, M.; Rump, F.J. (eds.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings of the GI-Workshop EPK 2003, pp. 31-44.

[Be01] Beech, D.; Lawrence, S.; Moloney, M.; Mendelsohn, N.; Thompson, H.S. (eds.): XML Schema Part 1: Structures. World Wide Web Consortium, Boston, USA, 2001. http://w3c.org/TR/2001/REC-xmlschema-1-20010502/.

[Bi95] Biskup, J.: Achievements of relational database schema design theory revisited. In: Libkin, L.; Thalheim, B.: Semantics in Databases, LNCS 1358, 1998, pp. 29-54.

[Bi03] Billington, J.; Christensen, S.; van Hee, K.E.; Kindler, E.; Kummer, O.; Petrucci, L.; Post, R.; Stehno, C.; Weber, M.: The Petri Net Markup Language: Concepts, Technology, and Tools. In: van der Aalst, W.M.P.; Best, E. (eds.): Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003, Eindhoven, The Netherlands, June 23-27, 2003, Proceedings. LNCS 2679, 2003, pp. 483-505.

[BM01] Biron, P.V.; Malhotra, A. (eds.): XML Schema Part 2: Datatypes. World Wide Web Consortium, Boston, USA, 2001. http://w3c.org/TR/2001/REC-xmlschema-2-20010502/.

[Bo98] Bos, B.; Lie, H.W.; Lilley, C.; Jacobs, I. (eds.): Cascading Style Sheets, level 2 – CSS2 Specification. http://w3c.org/TR/CSS2, 1998.

[BO02] Brabänder, E.; Ochs, H.: Analyse und Gestaltung prozessorientierter Risikomanagementsysteme mit Ereignisgesteuerten Prozessketten. In: Nüttgens, M.; Rump, F.J. (eds.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings of the GI-Workshop EPK 2002, pp. 17-34.

[CD99]   Clark, J.; DeRose, S.: XML Path Language (XPath) Version 1.0, World Wide Web Consortium, Boston, USA, 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[Ch01]   Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S.: Web Service Description Language (WSDL) 1.1, World Wide Web Consortium, Boston, USA, 2001. http://www.w3.org/TR/wsdl.

[Cl99]   Clark, J. (ed.): XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium, Boston, USA, 1999. http://w3c.org/TR/1999/REC-xslt-19991116/.

[CS94]   Chen, R.; Scheer, A.-W.: Modellierung von Prozessketten mittels Petri-Netz-Theorie, in: Scheer, A.-W. (ed.): Publications of the Institut für Wirtschaftsinformatik, No. 107, Saarbrücken 1994.

[DC03]   Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1: Reference Description. 2003. http://dublincore.org/documents/2003/02/04/dces/.

[De02]   Dehnert, J.: Making EPC´s fit for Workflow Management, in: Nüttgens, M.; Rump, F.J. (eds.): Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten - EPK 2002, Proceedings of the GI-Workshop EPK 2002, Trier, 2002, pp. 51-69.

[De03]   Delphi Group (ed.): BPM 2003 – Market Milestone Report. Delphi Group White Paper. Boston, 2003.

[EM01]   Embley, D.W.; Mok, W.Y.: Developing XML documents with guaranteed "good" properties. In: Kunii, H.S.; Jajodia, S.; Sølvberg, A. (eds.): Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling, LNCS 2224, 2001, pp. 426 – 441.

[Fe03]   Ferraiolo, J.; Jun, F.; Jackson, D. (eds.): Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3c.org/TR/SVG11, 2003.

[Fi92]   Finkelstein, A.; Kramer, J.; Nuseibeh, B.; Finkelstein, L.; Goedicke, M.: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. In: International Journal of Software Engineering and Knowledge Engineering. 2 (1992) 1, pp. 31-57.

[Ga02]   Gartner Research: The BPA Market Catches Another Major Updraft. Gartner's Application Development & Maintenance Research Note M-16-8153, 12 June 2002.

[IDS01]   IDS Scheer AG (ed.): XML-Export und-Import mit ARIS 5.0, Stand Januar 2001, Saarbrücken, 2001.

[IDS03]   IDS Scheer AG (ed.):ARIS Process Performance Manager, Whitepaper 2003, Saarbrücken.   www.ids-scheer.com/sixcms/media.php/ 1186/aris_ppm_whitepaper_e_v500.pdf.

[ISO01]   Ketels, K.: ISO 15022 XML Design Rules, Technical Specification, 2001. http://xml.coverpages.org/ISO15022-XMLDesignRulesV23a.pdf.

[Ke99]   Keller, G. & Partner: SAP R/3 prozessorientiert anwenden. Iteratives Prozeß-Prototyping mit Ereignisgesteuerten Prozeßketten und Knowledge Maps, Bonn et al. 1999.

[Ki03]   Kindler, E.: On the semantics of EPCs: A framework for resolving the vicious circle (Extended   Abstract).   In: Nüttgens,   M.;   Rump,   F.J.   (eds.):   EPK   2003   -

Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings of the GI-Workshop EPK 2003, pp. 7-18.

[KK02]   Karagiannis, D.; Kühn, H.: Metamodelling Platforms. Invited Paper. In: Bauknecht, K.; Min Tjoa, A.; Quirchmayer, G. (eds.): Proceedings of the 3rd International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, Springer-Verlag, p. 182-196.

[KM94]   Keller, G.; Meinhardt, S.: SAP R/3-Analyzer: Optimierung von Geschäftsprozessen auf der Basis des R/3-Referenzmodells, Walldorf, 1994.

[KNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)". In: Scheer, A.-W. (eds.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Saarbrücken, 1992.

[LSW98] Langner, P.; Schneider, C.; Wehler, J.: Petri Net Based Certification of Event driven Process Chains, in: Desel, J.; Silva, M. (eds.): Application and Theory of Petri Nets 1998, LNCS Vol. 1420, Springer, Berlin et. al. 1998, pp. 286-305.

[Mi03]   Microsoft (ed.): About the XML for Visio Schema. MSDN Library, 2003. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devref/HTML/XMLR_XMLBasics_818.asp

[MN02]   Mendling, J.; Nüttgens, M.: Event-Driven-Process-Chain-Markup-Language (EPML): Anforderungen zur Definition eines XML-Schemas für Ereignisgesteuerte Prozessketten (EPK). In: Nüttgens, M.; Rump, F. (eds.): EPK 2002 – Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings of the GI-Workshop EPK 2002, pp. 87-93.

[MN03a] Mendling, J.; Nüttgens, M.: EPC Modelling based on Implicit Arc Types. In: Godlevsky, M.; Liddle, S.W.; Mayr, H.C.: Proc. of the 2nd International Conference on Information Systems Technology and its Applications (ISTA), LNI Vol. P-30, Bonn 2003, pp. 131-142.

[MN03b] Mendling, J.; Nüttgens, M.: XML-basierte Geschäftsprozessmodellierung. In: Uhr, W., Esswein, W.; Schoop, E. (eds.): Wirtschaftsinformatik 2003 / Band II, Heidelberg, 2003, pp. 161 -180.

[MN03c] Mendling, J.; Nüttgens, M.: EPC Syntax Validation with XML Schema Languages. In: Nüttgens, M.; Rump, F.J. (eds.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings of the GI-Workshop EPK 2003, pp. 19-30.

[MN04]   Mendling, J.; Nüttgens, M.: XML-based Reference Modelling: Foundations of an EPC Markup Language. In: Becker, J. (ed.): Proceedings of the 8th GI Workshop „Referenzmodellierung", Essen, Germany.

[NR02]   Nüttgens, M.; Rump, J.F.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: Desel, J.; Weske, M. (eds.): Promise 2002 - Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen, Proceedings GI-Workshop und Fachgruppentreffen (Potsdam, Oktober 2002), LNI Vol. P-21, Bonn 2002, pp. 64-77.

[NS02]   Neumann, G.; Strembeck, M.: A scenario-driven role engineering process for functional RBAC roles. In: 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002), pp. 33-42.

[OMG03]Object Management Group (ed.): XML Metadata Interchange (XMI) Specification, May 2003, Version 2.0, 2003.

[Ös95]   Österle, H.: Business Engineering. Prozess- und Systementwicklung, Band 1, Entwurfstechniken, Berlin, 1995.

[OW04]   The OWL Services Coalition (ed.): OWL-S: Semantic Markup for Web Services. Whitepaper Version 1.0. http://www.daml.org/services, 2004.

[Ri00]   Rittgen, P.: Paving the Road to Business Process Automation, European Conference on Information Systems (ECIS) 2000, Vienna, Austria, July 3 - 5, 2000, pp. 313-319.

[RM98]   Rosemann, M.; zur Mühlen, M.: Evaluation of Workflow Management Systems - A Meta Model Approach. In: Australian Journal of Information Systems 6 (1998) 1, pp. 103-116.

[Ru99]   Rump, F.: Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten - Formalisierung, Analyse und Ausführung von EPKs, Teubner, Stuttgart et al. 1999.

[Sc00]   Scheer, A.-W.: ARIS business process modelling, Berlin et al., 2000.

[SW01]   SWIFT (ed.): SWIFTStandards XML Design Rules Version 2.3, Technical Specification, 2001. http://xml.coverpages.org/EBTWG-SWIFTStandards-XML200110.pdf.

[Wf02]   Workflow Management Coalition (ed.): Workflow Process Definition Interface – XML Process Definition Language, Document Number WFMC-TC-1025, October 25, 2002, Version 1.0, Lighthouse Point, USA, 2002.

[Wh03]   White, S.A: Business Process Modeling Notation – Working Draft 1.0, Aug. 25, 2003. BPMI.org, 2003.

[WHB02]Wüstner, E.; Hotzel, T.; Buxmann, P.: Converting Business Documents: A Classification of Problems and Solutions using XML/XSLT. In: Proceedings of the 4th International Workshop on Advanced Issues of E-Commerce and Web-based Systems (WECWIS 2002).

[WK02]   M. Weber, E. Kindler: The Petri Net Markup Language. In: H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber (eds.): Petri Net Technology for Communication Based Systems. LNCS 2472, 2002.

[WKR02]Winter, A.; Kullbach, B.; Riediger, V.: An Overview of the GXL Graph Exchange Language. In: Diehl, S. (ed.) Software Visualization - International Seminar Dagstuhl Castle, LNCS 2269, 2001.