XML-based Reference Modelling: Foundations of an EPC Markup Language

Jan Mendling, Markus Nüttgens

The advent of XML has forced the vendors of Business Process Modelling (BPM) tools to include respective import and export interfaces in their packages. But in order to leverage the benefits of XML model interchange, standardised vocabularies have to be developed. This paper describes the proposal of an EPC Markup Language from its guiding design principles to its concrete definition. We gather findings from other XML standardisation initiatives and derive general EPML design principles, as well as theoretical and practical XML design guidelines. A survey on graph representation in XML languages founds the decision to model EPC processes as edge element lists. Subsequently, the syntactical elements of EPML describing EPC hierarchies, EPC control flow, graphical display of objects, and business perspectives on EPCs are discussed.

1 Interchanging Business Process Models

In the 1990s tools for Business Process Modelling (BPM) have grown to become a software market segment of its own. In 2002 Gartner Research has been expecting a consolidation which only half of the 35 major tool vendors will eventually survive [Gart02]. As a consequence, interoperability and use of standards are becoming a major sales pitch. BPM tool vendors rely on XML technology to meet these new requirements. There is a growing number of products supporting XML import and export of business process models [MeNü03b]. In this context two levels of XML support have to be distinguished. On the one hand, we will refer to usage of XML as a standardized representation format of tool specific content as weak standard support. On the other hand, the usage of XML-based interchange standards will be referred to as strong standard support.

Weak standard support describes a strategy of BPM tool vendors to provide XML interfaces which correspond to a proprietary XML schema. This implies a greater transparency of the data stored and a sales pitch due to XML standard support. Concerning integration this is not a real progress. In a heterogeneous environment of different tools all providing weak standard support, transformation programs are still required in order to edit a model designed with *tool a* in *tool b*. There is an advantage contributed by the common use of XML: XSLT [Clar99] provides a scripting language for transformations that simplifies parsing of the input file,

specification of mapping rules, and assembly of the output. But this does not reduce heterogeneity of tools.

Strong standard support implies the existence of a standardized XML schema which is supported by tool vendors as an import and export format. This is an efficient situation for the user: she may use different tools for different purposes and interchange the models via an XML file conforming to the standardized XML schema. Such standards have been established for modelling methodologies like the Unified Modeling Language (UML) [OMG03a] in shape of XML Metadata Interchange (XMI) [OMG03b] and for Petri Nets with Petri Net Markup Language (PNML) [Bisk03]. For BPM with Event-Driven Process Chains (EPC) [KeNS92] such a specification is in progress of development. It is called EPC Markup Language (EPML) [MeNü02; Mend03; MeNü03b; MeNü03c]. The establishment of a standardized representation of business process models may be even more beneficial than in other domains, because interchange may have 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 [WfMC02]. This is a crucial step to finally close the engineering gap between modelling and implementation.

Today, the BPM market has adopted weak standard support providing XML interfaces for their tools. The development of EPML may eventually encourage BPM tool vendors to choose a strategy of strong standard support. Meanwhile EPML can be used as an intermediary format. With a large number of tools it is beneficial to use such an intermediary in order to reduce the number of transformation scripts and limit the loss of information [WHBC02]. The development of interchange formats for business process modelling has a significant impact on reference modelling. Once such an interchange format is standardized and accepted, reference models can be exchanged and reused beyond system and tool boundaries. This paper will discuss how an intermediary format for EPCs can be designed. Section 2 addresses general XML design principles and design principles for EPML in particular. Section 3 deals with process and graph representation in XML. Best practices will be extracted from different graph-oriented markup languages. Section 4 will be dedicated to EPC process graph objects. In this context EPC syntax elements and their logical relationships will be discussed. In section 5 the representation of graphical aspects are examined including coordinate system, position and layout information. Section 6 will provide a survey on business perspectives, views and dimensions related to BPM in an organizational environment. Each of the discussions in the various sections will conclude with design proposals for EPML. Section 7 will present a summary of the findings and an outlook on EPML.

2 EPML Design Principles

The purpose of EPML is to provide a tool and platform independent XML-based interchange format for EPCs. This mainly implies three questions: firstly, the question arises of *what* shall be modelled in details. Secondly, there have to be general guidelines on *how* things have to be expressed in XML. Thirdly, there is the question of *which* general principles shall guide the modelling. This section will begin the discussion with the *which-principles-question* and continue with the *how-question*. The question concerning details will be captured in the following sections.

2.1 EPML General Design Principles

Some of the various XML specifications of special domain vocabularies explicitly describe their general design principles. One of them is the ASC X12 Reference Model for XML Design (X12) [ANSI02] that describes 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. XML design guidelines are affected by this principle.

The PNML approach for Petri Nets is governed by the principles flexibility, no ambiguity, and compatibility [BCHK03]. *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 [MeNü03b]. *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. Graphical Information is discussed in section 5. Finally, *syntactical correctness* summarizes aspects dealing with EPC syntax elements and their interrelation. This principle is related to sections 3 and 4. The following paragraph will discuss general XML design aspects.

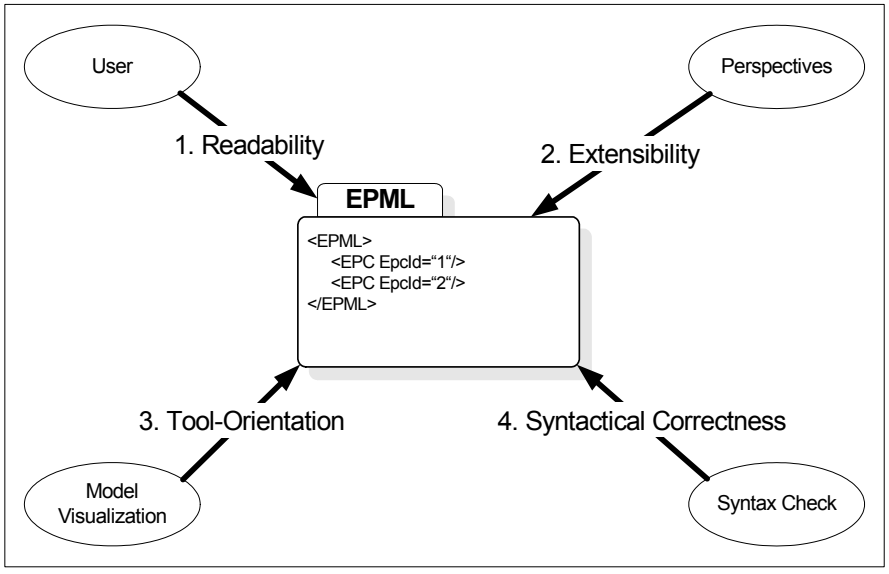


Figure 1: EPML Design Principles

2.2 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), multivalue dependency (MVD), and join dependency (JD) have been formally described [Bisk95]. 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) [EmMo01; ArLi02]. In [ArLi03] 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 [ArLi03]. 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 [ArLi03]. 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 [ArLi03]. For XML reference model design this implies that there should be no XPath [ClDe99] 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 [Kete01], SWIFT [SWIF01], MISMO [MISM02] and X12 [ANSI02] 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 [ANSI02]. 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 [BLMM01; BiMa01]. Further, attributes are considered to provide a better readability of content [Mert01; ANSI02]. 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 [ANSI02] 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. The following section continues with an analysis of process graph representation in XML.

3 Process Graph Representation

A graph is a pair of vertices V and edges E with E being a subset of the Cartesian product of V. Graphs can be found in various domains of computer science. For example, Entity-Relationship-Diagrams are used as conceptual representation in relational database design [Chen76]. Entities can be regarded as special vertices and relationships as special edges. Another example is object-oriented software engineering. The Unified Modeling Language (UML) [OMG03a] allows relationships and inheritance hierarchies to be modelled which can be interpreted as graphs. Graph-like structures of software programs are retrieved and rearranged in software reengineering [FaGW03]. As business process modelling formally builds upon directed graphs, an approach towards a XML representation for EPCs will have to take insights from these domains into account.

In computer science various data structures for graphs are discussed, mainly with focus on the efficient execution of graph algorithms. The three most prominent of them are adjacency matrices, adjacency lists, and edge lists [Eber87]. A adjacency matrix represents a directed graph with n vertices using an $n \times n$ matrix, where the entry at (i,j) is 1 if there is an edge from vertex i to vertex j; otherwise the entry is 0 [Blac03]. In contrast adjacency lists describe directed graphs with n vertices using an array of n lists of vertices. A vertex j is included in list i if there is an edge from vertex i to vertex j. Edge lists come closest to the set-oriented definition of graphs. An edge for a vertex i to a vertex j is stored as a pair (i,j).

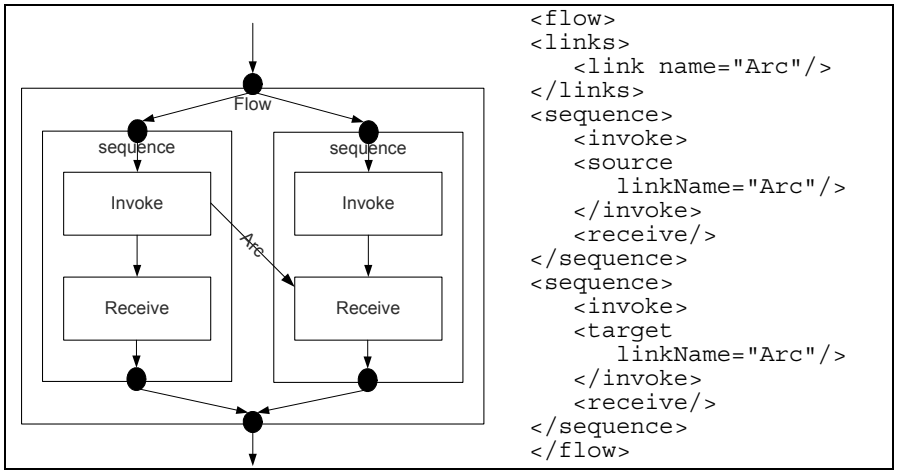
When such a generic graph data structure shall to be expressed in XML, adaptations have to be made taking the tree-like structure of XML into consideration. This implies that in general ID, IDREF, IDREFS data types known from Document Type Definitions (DTD) [BPSM00] or xs:key, xs:keyref constraints from XML Schema [BLMM01; BiMa01] have to be used to express arbitrary edges. In order to identify best practices in expressing graphs in XML, we will have a look at eight different XML graph representations, including

- AML, the XML format of ARIS Toolset [IDS01; IDS03a];
- The Business Process Modeling Language (BPML) proposed by BPMI.org, an industry initiative of companies dedicated to BPM [Arki02];

- The Business Process Execution Language for Web Services (BPEL4WS) promoted by IBM, Microsoft, BEA, Siebel, and SAP [ACDG03];
- The Graph eXchange Language (GXL), a specification of the software reengineering community [WiKR02];
- The Petri Net Markup Language (PNML) developed within the Petri Net community [WeKi02];
- MS Visio's VDX format allowing XML storage of Visio diagrams [Micr03];
- XML Metadata Interchange (XMI) from Object Management Group (OMG), the standard for exchanging UML models [OMG03b]; and
- XML Process Definition Language (XPDL), the proposal from Workflow Management Coalition (WfMC), the XML specification for WfMC's Interface 1 – process definition interchange [WfMC02].

These XML Schemas and DTDs come from academic proposals, industry standards, or tool-specific specifications. Their graph representation philosophies can be subdivided into three categories: block-oriented representation, adjacency subelement lists, edge element lists.

Block-oriented representation is used by novel business process modelling languages for Web Services like BPML or BPEL4Ws. This paradigm is inspired by process algebra like Pi-Calculus [Miln99] which serves as their theoretical foundation. Block-oriented languages provide a set of simple (in BPML) or basic (in BPEL4WS) and complex (in BPML) or structured (in BPEL4WS) operations that represent the control flow. There are some naming discrepancies between BPML and BPEL4WS, but the concepts are very similar [MeMü03]. Complex operations allow the definition of parallel execution, sequence, choices, and loops. They may be nested, but pure block structure is not able to express arbitrary control flows. Therefore, BPML and BPEL4WS include additional links to describe arbitrary synchronisation paths. It is an advantage of a block-oriented representation that code (without much nesting) is readable thanks to its sequential nature; and that only few commands are needed to express complex behaviour, compare figure 2. The disadvantage is that block orientation needs to mix with other concepts like links to express certain synchronisation behaviour; and that it is not meant for graphical presentation. Complex mappings are needed between modelling tools and block-oriented representation, as for example described in the Business Process Modelling Notation (BPMN) draft [Whit03]. EPML is meant for graphical BPM tools; therefore block-oriented representation of process graphs will not be used.




```
<flow>
<links>
   <link name="Arc"/>
</links>
<sequence>
   <invoke>
   <source
      linkName="Arc"/>
   </invoke>
   <receive/>
</sequence>
<sequence>
   <invoke>
   <target
      linkName="Arc"/>
   </invoke>
   <receive/>
</sequence>
</flow>
```

Figure 2: A process with two parallel paths and its abbreviated BPEL4WS syntax

Adjacency sub-element lists describe a process graph by an unordered list of nodes each having an ID attribute. An arc is represented by a sub-element of its source node. The arc carries an attribute which holds an ID reference to the node where it is linking to. The adjacency sub-element list representation is used by ARIS Toolset's XML export format [IDS01]. Its advantage is that you have a quick overview on which arcs leave from a certain node. But the use of ID references reduces readability. Another disadvantage stems from conceptual implications of this representation style: it is not possible to express arcs that do not have at least a beginning node. In BPM it may make sense to interchange process models that have not been finished yet containing arcs without start or end node. An adjacency sub-element list is not able to hold such information.

```
<node Id="1">
   <arc ToId="2"/>
   <arc ToId="3"/>
</node>
<node Id="2"/>
<node Id="3"/>
```

Figure 3a: Adjacency sub-element list representation. There are two arcs from node 1, one to node 2 and one to node 3.

```
<node Id="1"/>
<arc FromId="1" ToId="2"/>
<arc FromId="1" ToId="3"/>
<node="2"/>
<node="3"/>
```

Figure 3b: The respective process fragment in edge element list representation

Edge element lists are closely related to a set-oriented definition of graphs. Arcs are treated as first-class objects. Specifications like GXL and PNML underline this by attaching IDs to arcs just like to nodes. The edge element list representation is very popular. It is used by GXL, PNML, Visio's VDX, XMI, and XPDL. An asset of this representation is its flexibility. Arbitrary graphs can be described, and it is

even possible to store arcs that are (not yet) connected with nodes. A disadvantage is its usage of IDs and IDREFs thatwhich makes it difficult to read for humans.

The different process representation paradigms urge one to trade off the EPML design principles of readability and tool orientation. Readability is best supported by block-orientation because it does not use IDs and ID references. But graphical representation has to rely on complex mapping rules which contradicts tool orientation. Edge element lists are less readable, but very flexible data structure that is closely related to a set-oriented representation of process models. Another advantage is the fact that a lot of other specifications use them. This simplifies transformations to different tools and different methodologies. Therefore, edge element lists will be used to describe EPC process graphs in EPML.

4 Process Graph Elements and Their Relationships

In this section the EPML understanding of EPC control flow models will be presented. Business views and perspectives will be covered in section 6. As EPML builds on the concept of EPC Schema sets [NüRu02], it is possible to store more than one EPC model in an EPML file. First, an introduction is given to the organisation of multiple EPCs in an EPML file and the relationships which span beyond single EPC models. Afterwards, the elements of a single EPC model are explained in their EPML syntax.

4.1 Hierarchies of EPCs in EPML

<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 [DCMI03] 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 (see section 5). 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 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.

Table 1: High level elements of an EPML file

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>

In [NüRu02] 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 paragraph; 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 is presented in section 6.

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.

4.2 EPC Models in EPML Syntax

In this paragraph EPC syntax is covered. For an overview of EPC semantics related issues, we refer to [NüRu02] and [Kind03]. In [KeNS92] the EPC is introduced to represent temporal and logical dependencies in business processes. Elements of EPCs may be of function type (active elements) symbolized by <function>, event type (passive elements) represented by <event>, or of one of the three connector types AND, OR, or XOR which may be either split or join operators. The connectors are described by EPML elements <and>, <or>, and <xor>. These objects are linked via <arc> elements to express the control flow. Based on practical experience with the SAP Reference model, process interfaces and hierarchical functions had been introduced as additional element types of EPCs [KM94]. The <processInterface> is used to refer from the end of a process to a following process. A hierarchical <function> allows to define macro-processes with the help of sub-processes. Both kinds of relations are expressed by the help of a <toProcess> element whose @linkToEpcId represents the relation with another EPC process. Events, functions, process interfaces, connectors and control flow arcs are the syntactical elements of a so-called flat EPC Schema, the basic building block of an EPC Schema set [NüRu02]. They all share an @id attribute, a <name> element, a <description> element, a <graphics> element (described in section 5), and a <syntaxInfo> element which may cover information concerning implicit element types. Syntax information leverages the design principle of syntactical correctness and allows an easier verification of EPC syntax properties. For a discussion of implicit element types and EPC syntax properties we refer to [MeNü03a, MeNü03c].

Table 2: Control flow elements of an EPML file

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> ?

Some control flow objects have special elements. Potentially, the same events, functions, and process interfaces may be used multiple times in a hierarchy of EPCs. In order to avoid redundancy their respective XML tags may contain

<reference> elements instead of <name> and <description>. Such a reference refers to a definition of an event, a function, or a process interface that centrally store the name and the description. Functions also may have a <unit-Reference>. This refers to a business perspective and will be explained in section 6. Arcs have to connect two other control flow elements according to the edge element list representation. This is the purpose of the <flow> element. It contains two attributes which both refer to id-attributes of other control flow elements: @source and @target.

5 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 [BCHK03]. 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.

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) [BLLJ98]. 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.

Table 3: The graphics element of an EPML file

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

6 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 [FKNF92]. This approach has been extended for EPCs to allow a personalised presentation of a process model with perspectives of concern [BDFK03].

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 [Sche00]. The PROMET concept differentiates between business dimensions explicitly including organisation, data, functions, and personnel [Öste95]. An in-depth survey of organisational entities provided in workflow management systems is given in [RoMü98]. The link between role-

based access control (RBAC) and business scenarios is analysed in [NeSt02] and a methodology to generate role hierarchies is developed. From a delegation perspective [AaKV03] structure the organisational perspective of a workflow system into a meta model including resources, organisational units, users, and roles. In [Whit03] and [BeAN03] 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 [CCMW01] of Web Services as a new category of resource perspectives. Beyond resources there have been further perspectives proposed like e.g. risk [BrOc02], performance measurement [IDS03b] to name but a few.

Table 4: Business perspectives and views in EPML

EPML element	Attributes and Sub-Elements
<view>	@name <unit> * <unitRelation> *
<unit>	@unitId @name
<unitRelation>	@relationId @unitRef @subUnitRef @annotation ?
<unitReference>	@unitRef @role ? @value ?

The DAML-S Initiative is committed to the development of a standardised business process ontology for Web Service [DAML03]. 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 [KaKü02]. 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. 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.

7 Outlook on EPML

Throughout this paper we have presented our proposal for an EPC Markup Language (EPML). This approach is meant as an interchange format for EPC models. It follows the guiding principles of readability, extensibility, tool orientation, and syntactical correctness. Throughout the different sections, we discussed best practices from other graph and process reference models and made our design decisions explicit. This included a detailed discussion of process graph representation, EPC process graph elements and their relationships, graphical information as well as business perspectives and views.

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) [FeJJ03]. A similar script will be developed from EPML to SVG. Thirdly, an XSLT-based [Clar99] syntax checker will be developed and continue the efforts of an XML-based syntax validation of EPCs [MeNü03c]. 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. Up-to-date information, material, and discussion on EPML can be found at http://wi.wu-wien.ac.at/Wer_sind_wir/ mendling/EPML/.

8 References

[AaKV03] 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.

[ArLi02] 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.

[ArLi03] 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.

[ANSI02] ANSI (ed.): ASC X12 Reference Model for XML Design, July 2002. http://www.x12.org/x12org/comments/X12Reference_Model_For_XML_Design.pdf.

[ACDG03] 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.

[Arki02] Arkin, A.: Business Process Modeling Language (BPML). BPMI.org, 2002.

[BeAN03] Becker, J.; Algermissen, L.; Niehaves, B.: Prozessmodellierung in eGovernment-Projekten mit der eEPK. In: M. Nüttgens, F. J. Rump (eds.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Proceedings of the GI-Workshop EPK 2003, pp. 31-44.

[BLMM01] Beech, D.; Lawrence, S.; Moloney, M.; Mendelsohn, N.; Thompson, H. s. (eds.): XML Schema Part 1: Structures. World Wide Web Consortium, Boston 2001. http://w3c.org/TR/2001/REC-xmlschema-1-20010502/.

[BDFK03] Becker, J.; Delfmann, P.; Falk, T.; Knackstedt, R.: Multiperspektivische ereignisgesteuerte Prozessketten. In: M. Nüttgens, F. J. Rump (eds.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Proceedings of the GI-Workshop EPK 2003, pp. 45-60.

[Bisk95] Biskup, J.: Achievements of relational database schema design theory revisited. In: L. Libkin, B. Thalheim (eds.): Semantics in Databases, LNCS 1358, 1998, pp. 29-54.

[BCHK03] 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: W. M. P.van der Aalst, E. Best (eds.): Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003. Eindhoven 2003, pp. 483-505.

[Blac03] Black, P. E.: NIST Dictionary of Algorithms and Data Structures, 2003. http://www.nist.gov/dads/.

[BiMa01] Biron, P. V.; Malhotra, A. (eds.): XML Schema Part 2: Datatypes. World Wide Web Consortium, Boston 2001. http://w3c.org/TR/2001/REC-xml schema-2-20010502/.

[BLLJ98] Bos, B.; Lie, H. W.; Lilley, C.; Jacobs, I. (eds.): Cascading Style Sheets, level 2 – CSS2 Specification. http://w3c.org/TR/CSS2, 1998.

[BrOc02] Brabänder, E.; Ochs, H.: Analyse und Gestaltung prozessorientierter Risikomanagementsysteme mit Ereignisgesteuerten Prozessketten. In: M. Nüttgens, F. J. Rump (eds.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Proceedings of the GI-Workshop EPK 2002, pp. 17-34.

[BPSM00] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E. (eds.): Extensible Markup Language (XML) 1.0 (Second Edition). World Wide Web Consortium, Boston, USA, 2000. http://www.w3c.org/TR/2000/REC-xml-20001 006/.

[ClDe99] Clark, J.; DeRose, S.: XML Path Language (XPath) Version 1.0, World Wide Web Consortium. Boston 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[Chen76] Chen, P.: The Entity-Relationship Model – Towards a Unitied view of Data. ACM Transactions on Database Systems. 1 (1976) 1, pp. 9-36.

[CCMW01] Christensen, E.; Curbera, F.; Meredith, G.; Weerawarana, S.: Web Service Description Language (WSDL) 1.1, World Wide Web Consortium. Boston 2001. http://www.w3.org/TR/wsdl.

[Clar99] Clark, J. (ed.): XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium. Boston 1999. http://w3c.org/TR/1999/REC-xslt-19991 116/.

[DAML03] The DAML Services Coalition (ed.): DAML-S: Semantic Markup for Web Services. Whitepaper Version 0.9. http://www.daml.org/services, 2003.

[DCMI03] Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1: Reference Description. 2003. http://dublincore.org/documents/ 2003/02/04/dces/.

[Eber87] Ebert, J.: A Versatile Data Structure for Edge-Oriented Graph Algorithms. CACM. 30 (1987) 6, pp. 513-519.

[EmMo01] Embley, D.W.; Mok, W.Y.: Developing XML documents with guaranteed “good” properties. In: H. s. Kunii, S. Jajodia, A. Sølvberg (eds.): Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling, LNCS 2224, 2001, pp. 426-441.

[FeJJ03] Ferraiolo, J.; Jun, F.; Jackson, D. (eds.): Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3c.org/TR/SVG11, 2003.

[FaGW03] Favre, J.-M.; Godfrey, M.; Winter, A.: First International Workshop on Meta-Models and Schemas for Reverse Engineering - Workshop Description, to appear in: Proceedings Working Conference on Reverse Engineering (WCRE 2003), IEEE Computer Society, 2003.

[FKNF92] Finkelstein, A.; Kramer, J.; Nuseibeh, B.; Finkelstein, L.; Goedicke, M.: Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. International Journal of Software Engineering and Knowledge Engineering. 2 (1992) 1, pp. 31-57.

[Gart02] 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.

[IDS03a] IDS Scheer AG (ed.): Schnittstellen zu ARIS 6.0x / 6.1x / 6.2, Whitepaper, Saarbrücken 2003. www.ids-scheer.de/sixcms/media.php/1049/Uebersicht+Schnittstellen+ARIS+2003-07.pdf.

[IDS03b] IDS Scheer AG (ed.):ARIS Process Performance Manager, Whitepaper, Saarbrücken 2003. www.ids-scheer.com/sixcms/media.php/1186/aris_ppm_whitepaper_e_v500.pdf.

[Kete01] Ketels, K.: ISO 15022 XML Design Rules, Technical Specification, 2001. http://xml.coverpages.org/ISO15022-XMLDesignRulesV23a.pdf.

[Kind03] Kindler, E.: On the semantics of EPCs: A framework for resolving the vicious circle (Extended Abstract). In: M. Nüttgens, F. J. Rump (eds.): EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Proceedings of the GI-Workshop EPK 2003, pp. 7-18.

[KaKü02] Karagiannis, D.; Kühn, H.: Metamodelling Platforms. In: K. Bauknecht; A. Min Tjoa; G. Quirchmayer (eds.): Proceedings of the 3rd International Conference EC-Web 2002 - Dexa 2002, Aix-en-Provence, France, September 2002, LNCS 2455, p. 182-196.

[KeMe94] Keller, G.; Meinhardt, S.: SAP R/3-Analyzer: Optimierung von Geschäftsprozessen auf der Basis des R/3-Referenzmodells, Walldorf 1994.

[KeNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“. In: A.-W. Scheer (ed.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Saarbrücken, 1992.

[Mert01] Mertz, D.: Subelement contents versus tag attributes. IBM DeveloperWorks – XML Zone, Nov 2001. http://www-106.ibm.com/developerworks/xml/library/x-tipsub.html.

[Mend03] Mendling, Jan: Event-Driven-Process-Chain-Markup-Language (EPML): Anforderungen, Konzeption und Anwendung eines XML Schemas für Ereignisgesteuerte Prozessketten (EPK). In: H. Höpfner, G. Saake (eds.): Proceedings of the Students Program in Conjunction with the 10th Symposium "Datenbanksysteme für Business, Technologie und Web". Magdeburg 2003, pp. 48-50.

[Miln99] Milner, R.: Communicating and Mobile Systems: The π-Calculus. Cambridge 1999.

[Micr03] 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

[MISM02] Mortgage Bankers Association of America (MISMO) (ed.): MISMO XML Design Rules and Guidelines. Draft 2.0 RC3, 2002. http://www.mismo.org/mismo/docs/drftspc/mismoengguidelines.pdf.

[MeMü03] Mendling, J.; Müller, M.: A Comparison of BPML and BPEL4Ws. In: R. Tolksdorf, R. Eckstein (eds.): Proceedings of the 1st Conference "Berliner XML-Tage". Berlin 2003, pp. 305-316.

[MeNü02] 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.

[MeNü03a] Mendling, J.; Nüttgens, M.: EPC Modelling based on Implicit Arc Types. In: M. Godlevsky, S. W. Liddle, H. C. Mayr (eds.): Proceedings of the 2nd International Conference on Information Systems Technology and its Applications (ISTA), LNI Vol. P-30. Bonn 2003, pp. 131-142.

[MeNü03b] Mendling, J.; Nüttgens, M.: XML-basierte Geschäftsprozessmodellierung. In: W. Uhr, W. Esswein, E. Schoop (eds.): Wirtschaftsinformatik 2003/Band II. Heidelberg, 2003, pp. 161-180.

[MeNü03c] Mendling, J.; Nüttgens, M.: EPC Syntax Validation with XML Schema Languages. In: M. Nüttgens, F. J. Rump (eds.): EPK 2003 – Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Proceedings of the GI-Workshop EPK 2003, pp. 19-30.

[NüRu02] Nüttgens, M.; Rump, J. F.: Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: J. Desel, M. Weske (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.

[NeSt02] 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.

[OMG03a] Object Management Group (ed.).: Unified Modeling Language (UML) Specification, Marc 2003, Version 1.5, 2003.

[OMG03b] Object Management Group (ed.): XML Metadata Interchange (XMI) Specification, May 2003, Version 2.0, 2003.

[Öste95] Österle, H.: Business Engineering. Prozess- und Systementwicklung, Band 1, Entwurfstechniken. Berlin 1995.

[RoMü98] Rosemann, M.; zur Mühlen, M.: Evaluation of Workflow Management Systems - A Meta Model Approach. Australian Journal of Information Systems 6 (1998) 1, pp. 103-116.

[Sche00] Scheer, A.-W.: ARIS business process modelling, Berlin et al., 2000.

[SWIF01] SWIFT (ed.): SWIFTStandards XML Design Rules Version 2.3, Technical Specification, 2001. http://xml.coverpages.org/EBTWG-SWIFTStandards-XML200110.pdf.

[WfMC02] 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 2002.

[Whit03] White, S.A: Business Process Modeling Notation – Working Draft 1.0, Aug. 25, 2003. BPMI.org, 2003.

[WüHB02] 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).

[WeKi02] 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.

[WiKR02] Winter, A.; Kullbach, B.; Riediger, V.: An Overview of the GXL Graph Exchange Language. In: s. Diehl (ed.): Software Visualization - International Seminar Dagstuhl Castle, LNCS 2269, 2001.