Representing block diagrams with XML: an application for production flow specification in Workflow area

V. CARCHILOLO, A. LONGHEU, M. MALGERI
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni - Università di Catania
V.le A. Doria, 6 – 95100 Catania – ITALY – Tel: +39-095-7382359 – Fax: +39-095-7382397

Abstract: Due to its flexibility, XML has been used to develop application-specific languages, as MathML or ChemML. In addition to represent the structure of information through XML tags, also appearance can be specified through XSL style sheets. In this paper, a very simple block diagram description language is defined, together with its representation with XSL. Then, an application case study is presented, considering production flows in workflow area and showing how its specification can be managed in a web-based environment using XML, XSL and Java applets.

Key-Words: XML, workflow, Java, specification language

1 Introduction
WWW is commonly considered as the richest data source, easily browsable thanks to its web of links among different pages. Behind this success, the main limitation when accessing web data is the lack of explicit structure, whose presence could help in understanding information semantics [1][2][3][4]. In this scenario, XML revolution [5] represents a powerful and simple way to give structure to data, improving data representation, comprehension and exchange. The basic idea underlying XML is to give tags on data the role of identifying their meaning rather than specifying how data should be represented, as HTML. The potential impact is significant, since servers and applications using XML can make their data available in a format highly simple and interoperable. Moreover, XML framework includes XSL [6], a formatting specification language through which it is possible to specify how a given XML document should be represented (on the web or other media), clearly separating information content from rendering making it easy to provide different views to the same data.

The flexibility of XML lies in the freedom of specifying one’s tags set, even if not only everyone can provide a new set of tags to represent his information, but also people inside a specific working group can define a common set of tags to describe their data. Such initiatives lead to develop application-specific languages using XML, as MathML [7] or ChemML [8]. In this paper, one of such languages is proposed. The idea is to describe data flow diagrams, used to represent the flow of something, form data to physical actions, in a specific context; for instance, data flow diagrams can be used to describe the flow of data inside an information system, or the sequence of actions that make up a recipe, or several steps to carry out an algorithm, and so on.

The XML-based language we propose is simple and general, and allows to represent simple actions (e.g. input and output operations), as well as structured blocks, as if, while, repeat and for constructs. We first describe such language, introducing the DTD where the set of tag is defined, then presenting an example of an application case study, that is production flows specification in workflow area.

2 XML and block diagrams: specification and architecture
In the following, a simple language for representing block diagrams is introduced. The idea is to provide a simple set of XML tags in order to allow the representation of a flow of data or actions.

Since we want to provide a general language, blocks introduced represent:
- a generic input, output, as well as processing operation, used as basic building blocks;
- structured blocks, as if, while, repeat and for constructs, used to empower flow representation.

The DTD where such blocks are defined is shown in Table 1. An XML document conforming to this DTD (hence, representing a block diagram), must present a start and end blocks enclosing a sequence of blocks, where each block can be a simple in, out, or processing operation, or a structured block, as described in Table 1.
Each block present an identifier as attribute, in order to provide a sequence number available for external applications to manage the diagram. All basic blocks, as well as start and end blocks, are described as PCDATA allowing to use any syntax for instructions (in, out, processing) or for a title (start, end). For the same reason, cond, index, fstart, fend are also defined as PCDATA; they represent, respectively, the condition to be used inside a structured (if, while, or repeat) block, the name of the variable to be used as index inside a for block, and the expressions to determine start and end value for the index inside the for block.

```xml
<!ELEMENT BlockDiagram (start, Block+, end) >
<!ELEMENT Block ( in | out | operation | if | while | repeat | for )* >
<!ELEMENT start (#PCDATA)>
<!ATTLIST start num ID #REQUIRED >
<!ELEMENT in (#PCDATA)>
<!ATTLIST in num ID #REQUIRED >
<!ELEMENT out (#PCDATA)>
<!ATTLIST out num ID #REQUIRED >
<!ELEMENT operation (#PCDATA)>
<!ATTLIST operation num ID #REQUIRED >
<!ELEMENT if (cond, True?, False?)>
<!ATTLIST if num ID #REQUIRED >
<!ELEMENT cond (#PCDATA)>
<!ELEMENT True (Block+)>
<!ELEMENT False (Block+)>
<!ELEMENT while (cond, Body)>
<!ATTLIST while num ID #REQUIRED >
<!ELEMENT Body (Block+)>
<!ELEMENT repeat (Body, cond)>
<!ATTLIST repeat num ID #REQUIRED >
<!ELEMENT for (index, fstart, fend, Body)>
<!ATTLIST for num ID #REQUIRED >
<!ELEMENT index (#PCDATA)>
<!ELEMENT fstart (#PCDATA)>
<!ELEMENT fend (#PCDATA)>
<!ELEMENT end (#PCDATA)>
<!ATTLIST end num ID #REQUIRED >
```

Table1. DTD Definition

We note that this DTD is very general, so it may be limited to describe specific situations; in such cases, further element and/or attributes may be added, given that the DTD described before can be used as a good starting point for describing more complex block diagrams.

In order to produce a graphical representation of XML documents representing block diagrams, several XSL style sheets can be easily defined; the style sheet we defined (not represented here since it is not very significant) allows to represent XML files using a classical block diagram representation, i.e. mapping each block type to the corresponding image: start and end blocks as ellipses, in, out, and processing operations as rectangles, if block as a rhombus, and so on.

We also note that XML and XSL can just be used to describe a block diagram, but a sort of engine it is always needed if it is required to “run” the block diagram in some way, for instance generating and executing the corresponding algorithm in a given programming language. For this purpose, a possible architecture is shown in Fig. 1.

The Java applet is used to allow managing flow diagrams in a web-based environment, and it contains two significant modules, that are the XML docs manager, used to generate and read XML files using XSL files and the specification (e.g. provided inside the Java applet using a GUI environment), and the interpreter, used to parse XML files in order to convert such specification into the desired format, e.g. generating the algorithm in a given programming language; in general, the output of this interpreter is defined as a set of actions coming from XML specification. In other words, the XML documents manager and the interpreter form the executable counterpart of XML representation.

The choice of a Java applet make it easy to operate in a web-based environment; however, we expect future implementations where XML-aware browsers can perform XML documents validation, while with a lightweight plug-in a distributed, secure, and simple management of XML documents can be performed. Moreover, browsers should also allow the use of different XSL style sheets for the same set of XML documents, so the diagram can be represented differently, according to specific requirements. We finally note that the proposed architecture is general, hence it may be required some other module in order to satisfy application-specific needs.
3 A case study application: Production Flow description

A case study application in which the block diagram description could be useful is the production flow specification in the workflow area [9][10][11]. The production flow is the sequence of actions to be performed in order to get a final product starting from raw material, hence a generic production flow can be easily viewed as a flow diagram. Two main reasons can be provided for using XML flow diagram description introduced previously in such environment:

- to provide a complete sharing of all information about production flow specification inside the facility intranet, i.e. in a web-based environment, also solving interoperability questions, always present in large manufacturing facilities;
- to allow managing different user classes, for which the representation of the same production flow could be different depending on the specific user group (e.g. design, CAD/CAM, marketing…).

Since XML can be used just for model representation, and we also need a way to manage the production flow (both during its specification and its execution), we also developed a Java applet [11] that manages the model through managing corresponding XML files. The applet, in conjunction with XML files, provide a visual user environment where production flow is managed in an user-friendly way. In the following, the production flow is described (extract from [12] and [13]), together with its representation in XML and according with the architecture mentioned in sec.2.

A generic product, core of production process, has an associated production flow (also named route). The association between a product and its production flow is shown in Fig.2, where each operation is an abstract action consisting of a sequence (script) of events. This two level specification is used since generally operations represent high-level (abstract) actions, while events represent atomic actions directly linked to production machines. This two level specification is also presented here in order to show how even more complex flow diagram description can use the simple description introduced in sec. 2.

Two classes of operations can be identified, i.e. production and optionality operations. Production operation represent all operation containing events, and they are specified in three steps: type, template and default. A type specify the physical action that the operation will perform (e.g. distinguishing assembly from measure operations). Since a type is characterized by having some specific events in its scripts, providing a type means to specify the core of a script. The template specify completely the structure of the script for an operation, i.e. all events not provided in the type phase, but no value for parameters is set.

Finally, choosing a default we select a set of values for parameters. Type, template and default, usually used to model production flows [12], are considered here for the same reason event level has been considered. Optionality operation represent a decision block that, depending on a condition based on one or more expressions, executes different sets of operations (paths) or assigns different parameters to the same operation. Since it has just a control flow role, such operation contain no event inside. It can easily noted that production operation can be mapped into basic operations, while optionality operation correspond to the if block. When more complex production flows are required, other constructs (e.g. while) can be used.

Based on these considerations, an example of a product specification is shown in Table2.

As mentioned before, this example includes the block diagram introduced in sec. 2 but also includes other tags (hence, an extension to the DTD presented for block diagrams should be made; here, we do not present such extension).

The first tag (after the <product>) is the set of attributes characterizing the product itself, as for instance its name. Then, the production flow for that product is shown, noting that the num attribute is used here as a sequential sequence number increased of a variable amount for each new operation (other choices can however be made).
We also note that input and output operations are used here to model the entrance (and exit) of material into (out from) production machines, showing how different semantics can be associated to the same XML document; in this sense, the architecture proposed in sec. 2 (or other frameworks) should be used to give such context-specific semantics to flow diagram representation. Then, the operation 0100 present additional attributes to describe which type, template and default should be associated, while the script tag is used to refer to an external XML document which contains the list of events present in the script of the operation 0100; the reference is made through a simple XML-link placed inside the script tag. According to the description of product specification, the script should vary according to the type, template and default chosen, hence these information should be used in conjunction with script link in order to refer to the correct sequence of events. An alternative solution could be to have type, template and default as links to three different XML document, the first containing the set of core events, the second containing all remaining events, and the last with a set of values to be given to all events. This is to show how easy and flexible the use of XML can be. The environment in which such production flow is run can be for instance the architecture proposed in sec. 2. Referring to that proposal, the Java applet can be effectively used as a cross-platform tool where production flow can be managed, from its definition, e.g. made with a GUI, to its storage and exchange by using XML, to its representation through XSL style sheets, to its execution, using the interpreter to translate XML files into production machines commands sequence.

4 Conclusions
In this paper, a simple XML-based language for flow diagrams description has been introduced. It allows simple operations as well as structured blocks, as classical if, while, repeat and for
constructs. Then, a general architecture in which such XML documents could be used is presented, consisting of a Java applet where the flow diagram is created, stored as an XML document and eventually made executable through some specific conversion.

Then, a sample application to use flow diagram is presented: the production flow management in workflow area. The example introduced is more complex than the simple flow diagram, however this can be easily extended to support production flow particular specification, showing how general and flexible is the flow diagram XML specification here presented.

5 REFERENCES
[5] XML – www.w3.org/Xml