Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Middleware based on XML technologies for achieving true interoperability between PLC programming tools

**E. Estévez, M. Marcos, D. Orive,
F. López, E. Irisarri, F. Pérez**

*Automatic Control and Systems Engineering Department, University of the Basque Country
Bilbao, Spain, (e-mail: {elisabet.estevez, marga.marcos, dario.orive, fabian.lopez, edurne.irisarri, federico.perez}@ehu.es)*

Abstract: Industrial Process Measurement and Control Systems are used in most of the industrial sectors to achieve production improvement, process optimization and time and cost reduction. Integration, reuse, flexibility and optimization are demanded to adapt to a rapidly changing and competitive market. In fact, standardization is a key goal to achieve these goals. The international standardization efforts have lead to the definition of the IEC 61131 standard. Part 3 of this standard defines a software model for defining automation projects as well as 5 programming languages. Nowadays, a major part of Programmable Logic Controllers (PLC) vendors follows this standard, although each programming tool adds particularities and stores the automation project in different manner. But, although they may use the same software model and the same programming languages, source code reuse is not available. This work presents an infrastructure that allows transferring source code from one PLC programming tool to any other transparently to the users. The proposal consists of a textual expression of the software model and the programming languages, as well as the mechanisms, based on XML technologies, to achieve tool interoperability.

1. INTRODUCTION

Nowadays most of the industrial sectors use Programmable Logic Controllers (PLCs) to achieve the control of their productive systems. In the last years, technological advances in these controllers allow the production improvement, process optimization and time and cost reduction. On the other hand, for many years, only proprietary programming languages could be used for vendor specific equipment. Although some languages, such as ladder diagram or instruction list were very widespread, their implementation used to be rather different. It was obvious the need of standardization in the field, covering from the hardware to configuration issues, up to the programming languages. In 1993, the International Electrotechnical Commission (IEC) published the IEC 61131, International Standard for Programmable Controllers (IEC, 2003).

The IEC 61131-3 standard deals with the software model and programming languages for Industrial Process Measurement and Control Systems (IPMCS) (Lewis, R.W, 1998), (John, K.H and Tiegelkamp M, 2001). In this sense, it has provoked a movement to Open Systems in this application field. Thus, the so-called Open PLCs that are open architecture controllers that replace a PLC with a computer, have begun to appear in the market.

Nowadays, most of the PLC vendors are doing a great effort for becoming IEC 61131-3 standard compliant. In fact, this offers great advantages to the control system engineers, as the programming techniques become vendor independent. Notwithstanding this, the standard does not specify an import/export format but the elements of the software model and the mechanisms to be offered to the user in order to graphically define an application. Thus, every tool uses its own storage format an offers commonly a set of Application Program Interface (API) functions or, alternatively, an import/export option. In this sense, it is impossible to reuse the code programmed in one tool in others. It is necessary to edit the code again. In order to achieve true reusability, interoperability among tools is needed. The international organization PLCopen (1992) is a vendor- and product-independent worldwide association, whose mission is to be the leading association resolving topics related to control programming. Its main goal is to support the use of international standards in this field. PLCopen has several technical and promotional committees (TCs). In particular, TC6 for XML has defined an open interface between all different kinds of software tools, which provides the ability to transfer the information that is on the screen to other platforms. The eXtensible Markup Language (XML) (W3C, 2006a) was selected for defining the interface format and in April 2004, the first XML schema for the graphical languages was released for comments (W3C, 2004).

Nevertheless, the PLCopen interface is not universally supported yet. Besides, the proposed interface focuses mainly on transferring what is in the screen and thus, adds graphical information as well as new elements to those used by the standard. Finally, it does not impose an architectural style, assuming that the code being transferred is correct. Thus the XML schema defined represents the elements of the IEC 61131-3 software model but it does not impose the restrictions that an automation project must meet. XML is being more and more used in factory automation (Younis and Frey, 2005), (Itoh et al., 2002), (Mizura, 2005), (E. Estevez,

et al, 2007a, b). The goal of the work presented here goes further, as an interoperability middleware is proposed. It consists of a common XML format for representing the IEC 61131-3 software model and languages and the mechanisms to import/export information from/to every tool. It is based on previous work of authors (Marcos and Estévez, 2005), (Estévez, et al., 2005) where a framework for integrating the tools involved in the development cycle of Industrial Control Systems is presented. This work focuses on the representation of the software architecture of the application defining the techniques that allow import and export information among tools.

The layout of the paper is as follows: section 2 briefly describes the elements of the IEC 61131-3 software model. Section 3 presents the interoperability middleware that acts as a common road for achieving true interoperability. Finally, section 4 illustrates an example of interoperability between two PLC programming tools.

2. THE IEC 61131-3 SOFTWARE MODEL

This section describes the software model proposed by IEC 61131-3 standard in order to identify the architectural style and composition rules that any application IEC 61131-3 compliant must meet.

The architectural style is identified in a Component-based fashion. The component-based strategy aims at managing complexity, shortening time-to-market, and reducing maintenance requirements by building systems with existing components. Generally speaking, software architectures are defined as configurations of components and connectors. An architectural style defines the patterns and the semantic constraints on a configuration of components and connectors.

In (E. Estevez, et al, 2007a) the different components and connectors for defining software model of IEC 61131-3 are identified. Two types of components can be distinguished: those that do not encapsulate code, Configurations, Resources and Tasks, and the Program Organization Unit (POU) that encapsulates code. These latter can be Programs, Function Blocks and Functions.

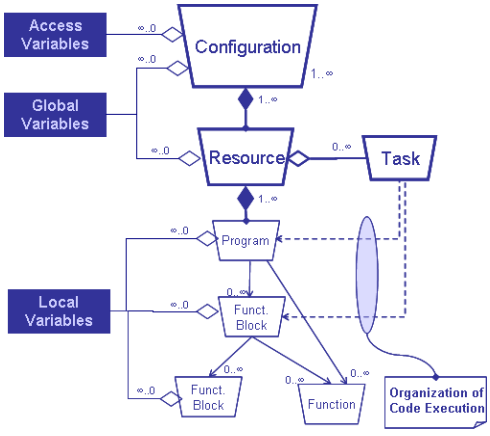

Fig. 1. Architectural style of IEC 61131-3 software model

On the other hand, in this model, the variables act as connectors. They represent the communication between software components. In fact, their visibility identifies the components that are involved in the communication. In Fig. 1 the architectural style of the IEC 61131-3 software model is illustrated using a meta-model expression. Every component and connector has its own characteristics as defined in (E. Estevez, et al, 2007a). This architectural is complemented with a set of composition rules that jointly defines the grammar that allows defining IEC 61131-3 software architecture. In Table 1 the identified composition rules are summarized.

TABLE 1. COMPOSITION RULES FOR IEC 61131-3 SW MODEL

Id	Rule
1	The type of a Global Variable must be elementary or user-defined
2	The value of Global Variables must be in concordance with their type
3	The type of POU formal parameters must be elementary or user defined
4	The value of POU actual parameters must be in concordance with their type
5	An Access variable must give permission to a previously defined variable
6	Resources of the same configuration must be downloaded to the same processor
7	Resource POU instances on the same resource must be organized by tasks of the same resource

3. INTEROPERABILITY MIDDLEWARE

The proposed interoperability middleware is formed by two main modules. The first one consists of a representation of the IEC 61131-3 software model in a standard and generic format. The integration of the model deals with mechanisms that allow interoperability among tools through the middleware, achieving code exchange.

The proposed framework is based on XML technologies. In particular XML schema (W3C, 2004) jointly with schematron rules (Rick, J., 2006) has been used for defining a common and generic format of the software model. This format takes into account both, the architectural style and the composition rules. Integration techniques involve related XML technologies, such as XML stylesheets and Document Object Model (DOM) (W3C, 2005) or Simple API for XML (SAX, 2004). Fig. 2 illustrates the general scenario of the proposed middleware for achieving true interoperability between any PLC programming tools.

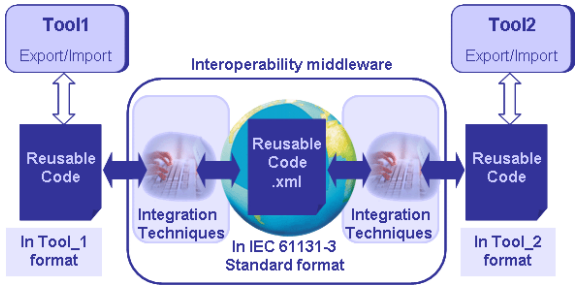

Fig. 2. General Scenario of interoperability among PLC programming tools

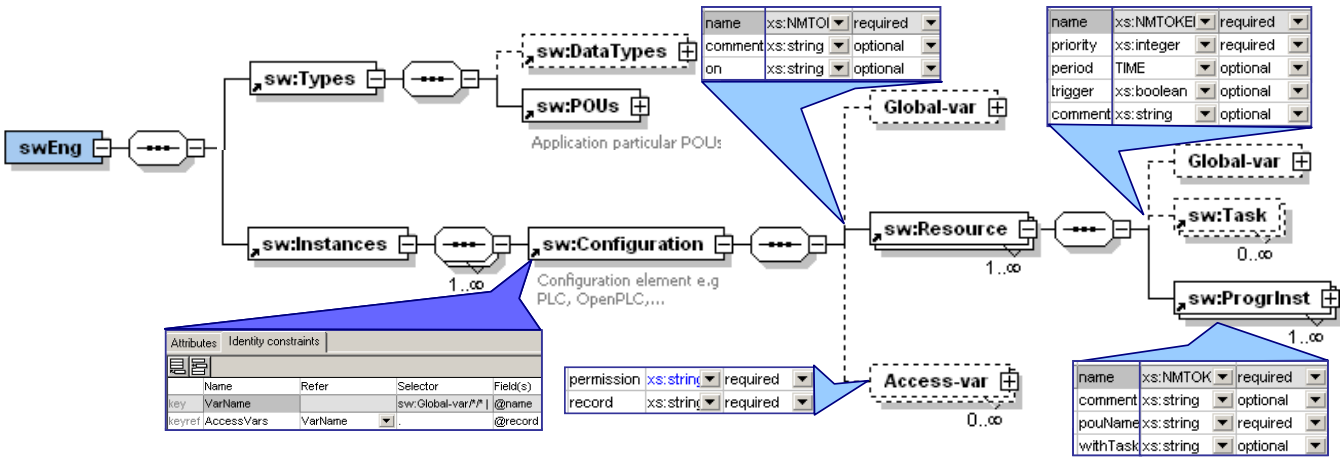

Fig. 3 Architectural style of IEC 61131-3 software model in swEng markup Language

The following sub-sections define both modules in more detail: a generic representation format of the software model and the integration mechanisms.

3.1. Standard format for CodeReuse

As commented above, the generic format proposed for representing the IEC 61131-3 software model, uses the W3C schema and schematron rules. In particular, each model element is represented as a XML schema element. The architectural style is performed making use of the choice, sequence and multiplicity mechanisms of W3C schema. On the other hand, the composition rules are performed by means of the *key/keyref* constraints of the W3C schema (Van der Vlist, E., 2002) and also by means of schematron rules. Fig. 3 illustrates a general overview of the IEC 61131-3 standard software model. It illustrates the architectural style as well as the implementation of composition rule number five of Table 1 that is implemented by means of the *key/keyref* mechanism. The complete definition of this module is available in (www.disa.bi.ehu.es/gcis/projects/merconidi).

3.2. Integration Techniques

In this section, the different integration techniques that allow transferring information between programming tools and the interoperability middleware are analyzed.

The integration techniques depend on the export/import format of the PLC programming tool, as well as on the API the tool offers. XML technologies offer powerful mechanisms for implementing tool integration.

Three tools categories can be distinguished:

a) Tools that import/export information in XML format, such as Multiprog™ from KW software that follows the PLCopen TC6 XML interface (www.kw-software.com/) or Unity Pro™ from Schneider (www.schneider-electric.com/).

b) Tools that import/export information in structured text format e.g. CoDeSys™ (www.3s-software.com) from 3S.

c) Tools that import/export information in any other format, e.g. ISaGRAF™ from ICS Triplex (www.isagraf.com) and Simatic Administrator™ from Siemens (www.automation.siemens.com).

Following sub-sections describe the selected integration technique for each type of PLC programming tool.

Tools with import/export option in XML format. If the tool provides XML interface or it allows exporting/importing projects to/from a XML file, the integration is practically direct. In this case, it is necessary to develop a XML stylesheet (W3C, 2006b). This XML technology can be used for processing an input XML file coming from the tool, filtering information and transforming it giving as output the reusable code expressed in the format proposed in this work (*tool2standard.xsl*). XSLT technology offers two types of templates that can be used to define the processing of the input file. The match template contains the processing to be applied to a particular XML element. This processing could be organized by means of the so-called name templates (Tidwell, D., 2001).The same XML technology is also used for filtering and transforming the reusable code expressed in standard form to the format of the target PLC programming tool (*standard2tool.xsl*).

In the first case the XSL *match* and *name* templates are tool dependent. The match templates of the second case are known; as they correspond to the elements of the IEC 61131-3 standard. Notwithstanding this, the algorithms they contain are again tool dependant. In Table 2 the necessary templates for transforming the reusable code in standard format to tool format are summarized.

Tools that import/export structured text. Although the number of tools that allows exporting/importing information in XML format is increasing, currently it is not the common. This sub-section describes the integration techniques for

those tools that allow exporting/importing code to/from structured text format. In this case, it is necessary to know the file structure.

TABLE 2. LIST OF *STANDARD2TOOL.XSL* TEMPLATES

Templates	Characteristics
match=”sw:swEng”	Main template, it guides de transformation.
<xsl:template match="sw:DataTypes">	Generates the derived data types
<xsl:template match="sw:POUs">	Organizes the POU structure
<xsl:template match="*" mode="POU">	Selects its type of POU
<xsl:template match="sw:Interface"> <xsl:template match="sw:Variables"> <xsl:template match="sw:Body"> <xsl:template match="sw:FBD"> …	A set of templates for generating the POU interface and functionality expressed in any of 5 languages of IEC 61131-3
<xsl:template match="sw:Configuration">	A set of templates for generating the automation project itself
<xsl:template match="sw:Resource"> <xsl:template name="globalVars"> <xsl:template match="sw:Task"> <xsl:template match="sw:ProgInst">	

There are different technologies that allow transforming structured text into XML, such as the Chaperon project (Stephan M, 2000). It can also be achieved by developing an application having as input file the structured text file and making use of DOM or SAX methods for generating an XML file. Finally, a XML stylesheet will be necessary for transforming this XML file to the standard format (see Fig. 4).

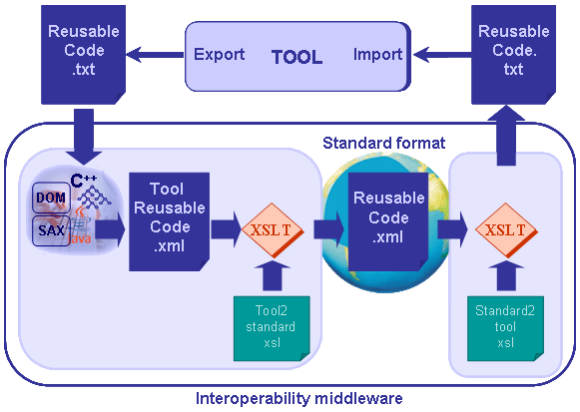

Fig. 4. General scenario for importing/exporting reusable code in structured text format

If the tool needs to import source code, XML stylesheets can be used to transform the code expressed in the generic XML format to a text file that follows the structure that the tool expects. Besides the templates of Table 2, it is also necessary to indicate to the XSLT processor the extension of the resulting text file. Fig. 4 illustrates the general scenario for exporting and importing reusable code.

Tools that import/export in any other format. Currently, this is the more common case, in which tools neither have an XML interface nor export/import code to/from structured text. In this case the integration technique consists of developing an application that makes use of functions provided by the tool API, as well as methods and functions offered by SAX or DOM.

Thus, the integration techniques for capturing information from the tool and to express it in a generic format consist of an application that generates an initial XML file and an XML stylesheet for filtering and transforming the information to the standard format.

The application programming language depends on the tool API and it also needs:

- The functions provided by the tool API for getting information from the automation project.
- The functions or methods provided by SAX or DOM. They are very useful for generating an initial XML file.

In the same way, the integration techniques for setting information coming from the generic XML file to the tool consist of an XML stylesheet that adds the tool particularities to the XML file. Furthermore, it is necessary an application that reads this XML file and sets all information in the storage format of the target tool. In this sense, the programming language of this application also depends on the tool API and it makes use of:

- SAX or DOM methods for reading and manipulating the input XML file
- Tool API functions for setting information into target tool storage format.

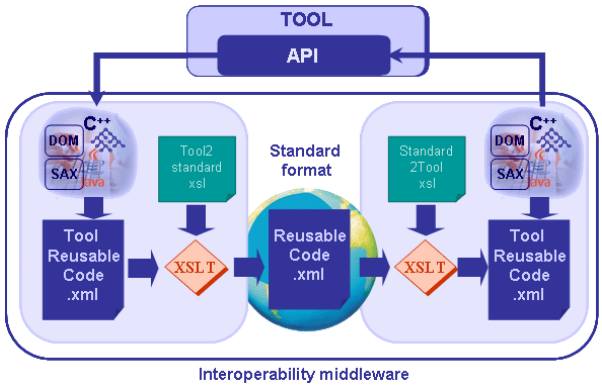

Fig. 5. General scenario for getting/setting information via tool API

Fig. 5 illustrates the general scenario for getting from the tool information via its API and transforming it to the generic format. In the same way this figure also illustrates the mechanisms for setting the code expressed in the standard format (*ReusableCode.xml*) to the target tool storage format.

4. CASE STUDY

This section illustrates the proposed interoperability framework as applied to transfer one POU programmed in CoDeSys™ from 3S to Multiprog™ from KW software. The first tool allows exporting/importing information to/from structured text format. The second follows the interface proposed by PLCopen TC6 XML. The code to be transferred

is a very simple example of a function (*inRange*) written in Function Block Diagram language. This POU checks if the content of a variable is between a minimum and a maximum. To do this, three standard functions have been used (LT, GT and AND). Fig. 6 illustrates this POU programmed in CoDesys™ PLC programming tool.

The first step, to achieve the interoperability is to export the code in text format. This definition contains the POU interface, formed by three input formal parameters, and the body expressed by reserved commands which represent the functionality, originally in FBD, in text format (note that this is a particularity of the CoDeSys™ tool).

In the second step (see Fig. 5), the interoperability middleware, an application programmed in java, reads this file using DOM, translates the textual file into an XML file. Then it applies an XML stylesheet to this file obtaining the *ReusableCode.xml* (illustrated in Fig. 7).

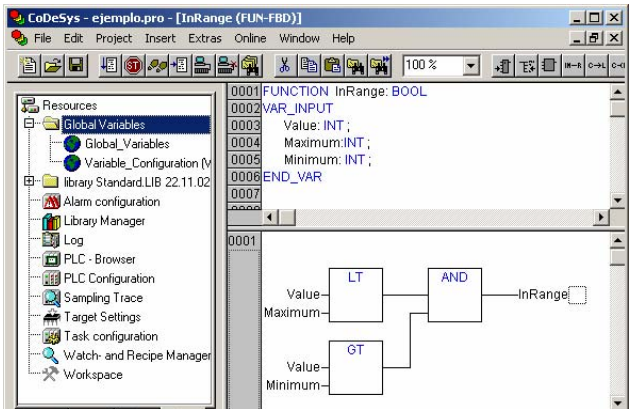

Fig. 6. POU example programmed in CoDeSys™

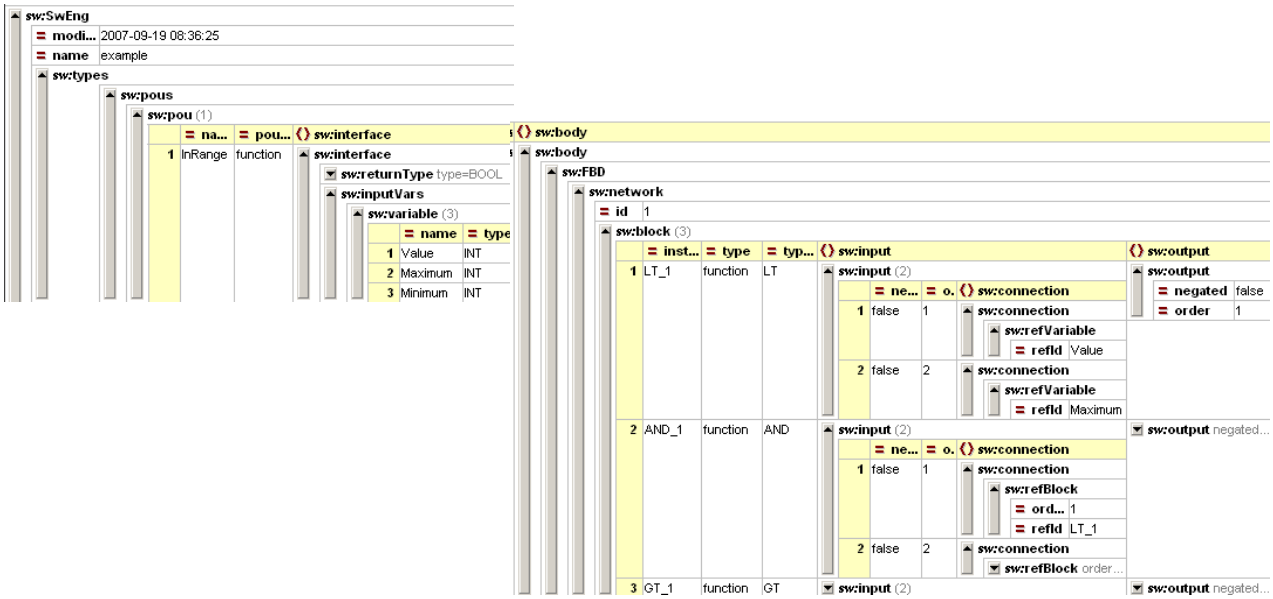

Fig. 7. Project expressed in generic format (*ReusableCode.xml*)

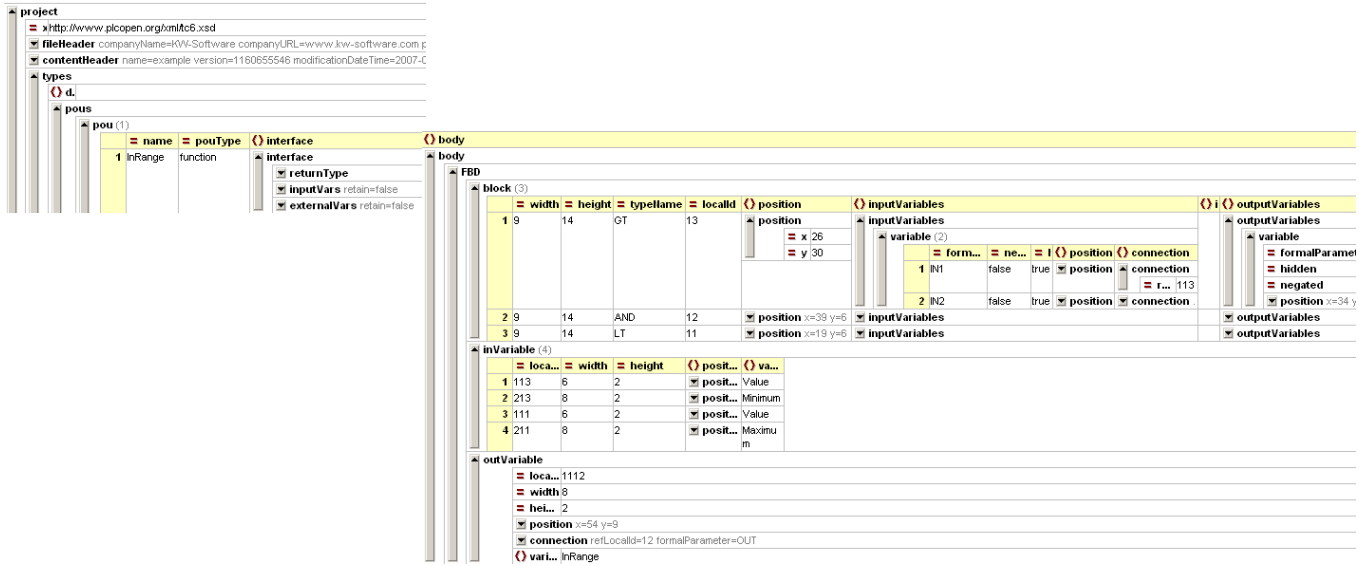

Fig. 8. Example in PLCopen TC6 XML format

The third step consists of transforming the generic format XML file to the PLCopen TC6 XML grammar. This transformation is done by means of a XML stylesheet. The resulting file of the transformation is illustrated in Fig. 8. This file contains graphical information, sometimes as attributes of a XML element (e.g. a block has the width and height attributes), and sometimes as a new element of the schema (e.g. a child element of a block is its position in the screen). Finally, the Multiprog™ tool can import this file. The resulting project is illustrated in Fig. 9.

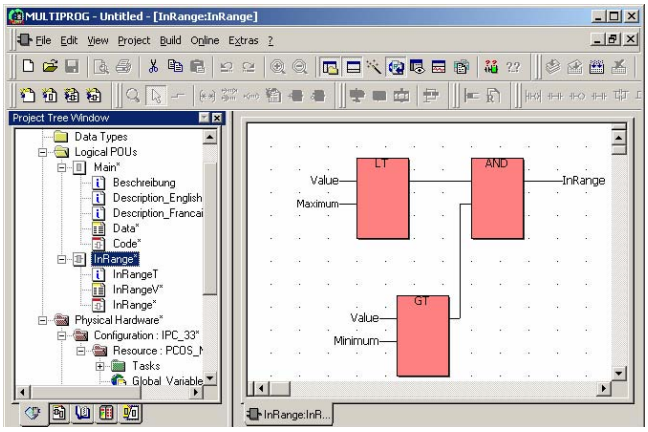

Fig. 9. InRange POU reused in Multiprog™ KW

Thus, by means of proposed integration middleware the *inRange* POU initially programmed in CoDeSys™ can be reused in Multiprog™ PLC programming tool. It is important to remark that the transformations, which are task of the middleware, are transparent to the user. It is necessary to develop the stylesheets that transform the software architecture and program languages between the PLC programming tool and the middleware framework. This will be as much simpler as higher the level of conformity with the IEC 61131 standard.

5. CONCLUSIONS

This paper has presented a formal approach that allows transferring source code among different vendors of PLC programming tools. Therefore, true interoperability between tools is achieved by means of the proposed middleware. The potentiality of XML technologies have been used for developing the interoperability middleware. In particular XML schema jointly with schematron rules form the core of the proposed middleware. They have been used to express in XML the IEC 61131-3 software model, taking into account the architectural style and composition rules. The middleware also offers mechanisms for tool integration making use of related XML technologies, such as XSLT, SAX and DOM. These techniques allows both transforming any tool model to a generic XML format, and obtaining a vendor understandable information from this generic XML format.

6. ACKNOWLEDGEMENTS

This work was financed by the MCYT&FEDER under DPI-2006-4003 and by UPV/EHU under project DIPE 06-16.

REFERENCES

E. Estévez, M. Marcos and D. Orive (2007a). *Automatic Generation of PLC Automation Projects from Component-Based Models*. The International Journal of Advanced Manufacturing Technology. Springer London. Vol: 35, pp: 527:540. 2007.

E. Estévez, M. Marcos, D. Orive, E. Irisarri and F. Lopez (2007b). *XML based Visualization of the IEC 61131-3 Graphical Languages*. Proc. of the 5th International Conference on Industrial Informatics, pp: 279-285 (INDIN 2007). Vienna, Austria.

E. Estévez, M. Marcos and U. Gangoiti, *"A Tool Integration Framework for Industrial Distributed Control Systems"* Proc. 44th IEEE Conference on Decision and Control and European Control Conference ECC'05. Seville.Spain.

IEC (2003). International Electrotechnical Commision. *IEC International Standard IEC 61131-3 Programmable Controllers. Par3: Programming Languages*.

John, K.H and Tiegelkamp M. (2001). *Programming Industrial Automation Systems.* Springer.

Lewis, R.W. (1998). *Programming Industrial Control Systems using IEC 61131-3.*IEE Control Engineering Series.

M. Marcos, E. Estévez, *"Formal Modelling of Industrial Distributed Control Systems"* 16th IFAC World Congress. Praha

M. Bani Younis, G. Frey, *"Formalization and Visualization of Non-Binary PLC programs"*, Proceedings of 44th IEEE Conference on Decision an Control, and the European Control Conference (CDC-ECC 2005), pp: 8367-8372. 2005

Mizura Tooru, *"XML representation for sequential control programs and its related software tool"*, Papers of Technical Meeting on Systems and Control, IEE Japan, pp: 7-10.2005.

PLCopen (1992), Web-site: http://www.plcopen.org

Rick J. (2006). *Resource Directory (RDDL) for Schematron 1.5.* Web Site: http://xml.ascc.net/schematron/

SAX (2004). *Simple API of XML (SAX).* Web Site: http://www.saxproject.org/

Stephan M. (2000), *Chaperon Project.* Web Site: http://chaperon.sourceforge.net/index.html

Tidwell, D.(2001). *XSLT*, Ed. O'REILLY.

Van der Vlist, E. (2002), *"XML Schema"*. Ed. O'REILLY.

W3C (2004). *XML Schema Part 0: Primer (Second Edition), W3C REC-xmlschema-0-20041028*.

W3C (2005). *Document Objecto Model (DOM)*, Web Site http://www.w3.org/DOM/

W3C (2006a). *eXtenslble Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation*. Available at: http://www.http://www.w3.org/TR/2006/REC-xml-20060816/

W3C (2006b). Extensible Stylesheet Language (XSL) Version 1.1, W3C Proposed Recommendation PR-xsl11-20061006.

Y. Itoh, M. Fukagawa, T. Nagao, T. Mizuya, I. Miyazawa, T. Sekipchi, *"On the interoperability and its test of software in the measurement and control domain"*. SICE 2002. Proceedings of the 41st SICE Annual Conference. Vol 1, pp:370-373.2002.