Advanced Materials Research *Online: 2014-10-01*
ISSN: 1662-8985, Vol. 1036, pp 770-774
doi:10.4028/www.scientific.net/AMR.1036.770

The Markup Language as a Form of Description of the Robot's Scene

Krzysztof Foit

Institute of Engineering Processes Automation and Integrated Manufacturing Systems

The Faculty of Mechanical Engineering, Silesian University of Technology,

Konarskiego 18A St, 44-100 Gliwice PL

E-mail address: krzysztof.foit@polsl.pl

Keywords: robotics, robot, programming, simulation, markup language

Abstract. The purpose of this paper is to show the possibilities offered by the use of markup language in the process of programming of the robot. In this paper, the author primarily focused on the problem of description of the objects' arrangement on the robot's scene. The XML format is selected among the various possibilities of using the markup language, due to the fact that most of the software for the simulation of manufacturing processes can export the virtual space configuration to the XML file. The XML (Extensible Markup Language) is a formal language that allows the representation of data structures in a way understandable to a human and suitable for machine processing. The other, significant advantage of the XML is that it is platform-independent language. Its structure is more understandable and much easier to learn than synthetic programming language, so it can be quickly acquired and used by a person who is not a programmer. In this paper, the focus has been mainly put on the presentation of the method and on the assumptions used in the examples given for a simple scene setup and simple manipulation tasks.

Introduction

A Modern industrial robot is seen as a machine capable of working in a diverse environment. This is manifested i.a. in the diversity of used effectors, variety of performed tasks, what is closely related to the configuration of the robot's workplace.

The immediate surroundings of the robot and the actions performed by the manipulator are reflected in the form of the robot's program. The program can be stored in a text or binary form, but the first one gives the portability of a code and the possibility to transfer it between different devices without using any converting software. In addition, there is no need to use special, dedicated software in order to make some changes inside the program. Eventually, the text form gives possibility of direct remarking, so the code is more understandable for the person, which is not directly engaged in the programming process. Unfortunately, the text form of the source code also has its drawbacks. For example, removing comments from code can make it completely incomprehensible and even the creator of the program – after some months or years later – may have trouble to explain what is the mean of particular functions or procedures – such cases are well known in the field of computer programming.

The ideal solution of the described problem would be the use of syntax of the code that is very close to natural language, but does not make any problems during further processing into machine-readable form. In fact, there is the need to seek a compromise between the possible human-intelligible form of description and its subsequent suitability for machine processing. This paper presents the proposal of the possible solution, which is based on markup languages. The markup languages are characterized by a specific form of writing, but its meaning can be easily understood by man without time-consuming study of the syntax rules. This approach gives the possibility of using human-readable form and ensures the existence of a wide range of machine processing capabilities.

The origins of the markup language

The idea of the markup language started with text processing. The word “markup” comes from “mark-up”, what means “to annotate” or “to comment”. The first appearance of the markup language progenitor is dated to 1967. It was the prototype language called “generic code” (“GenCode”), used for text processing, created by W. W. Tunnicliffe. Its main aim was to make the text processing independent of the target hardware (typesetting device), which often used specific control notation. This has given the life to SGML (Standard Generalized Markup Language) standard, created by ISO committee. However, the markup language has also the others “fathers”. The IBM employer and researcher C. F. Goldfarb, along with E. Mosher and R. Lorie has created the GML (Generalized Markup Language), initially for internal purposes of the company, but in 1973 it was presented to publicity. The GML is seen as a language that had a strong influence on the final form of the SGML [1].

The SGML has been the basis for creating other markup languages, including the two of today’s most popular: the HMTL (Hypertext Markup Language) and XML (Extensible Markup Language). First of them has been used for long time in order to format text displayed by web browsers, so its syntax is focused on text processing. The HTML has a lot in common with SGML and GML – it is also strictly specialized in word processing. The XML language has other nature. It is also document-oriented language, but not as strict as HTML, because the XML allows the user to create its own tags (markups). This flexibility causes, that it is willingly used as data storage format (for example Microsoft Office or OpenOffice have new documents formats, which are based on XML). The XML could also be used for presenting information from databases, giving the possibility to mix web-style interface with other data formats.

Tags and the overall structure of the SGML/XML document

The tag in general can be understood as a symbol or a label that gives us some kind of information. Tags are commonly used for identification and classification. For example, if somebody is organizing the conference, then he could prepare identifiers in different colors for organizers, participants and accompanying persons to introduce better organization and services. A tag could have different form – it could be a sign, word (or any sequence of characters), picture, flag etc. The examples from the everyday live could be vehicle registration plate, phone number, barcodes/QR codes, graphical symbols on different products (see Fig. 1) etc.

Fig. 1. Examples of tags in the form of pictograms placed on the air freshener container

Tags are the fundamental component of the SGML/XML markup language. They have the form of a word enclosed in angle brackets. Each tag has a corresponding closing one, which marks the end of data object. The value is placed between the tags. The structure of the XML document has the form of tree. The elements of hierarchy are [2]:

- root element,
- parent elements,
- child elements,
- sibling elements.

Every document has only one root element, but could contain any number of parent elements. The parent element could contain child elements. Any elements placed on the same hierarchy level are sibling elements. The hierarchy is graphically shown in Figure 2.

```
<root>
    <parent_1>
        <child_1>
            data
        </child_1>
        <child_2>
            data
        </child_2>
    .....
        <child_n>
            data
        </child_n>
    </parent_1>
    <parent_2>
        <child_1>
            data
        </child_1>
        <child_2>
            data
        </child_2>
    .....
        <child_n>
            data
        </child_n>
    </parent_2>
</root>
```

Fig. 2. Overall XML document hierarchy with two parent objects

Exact description of all XML features goes beyond the scope of this paper and could be found in the professional literature or dedicated websites.

Description of the robot scene in the XML language – general assumptions

The use of XML in robotics has gained importance together with the development of web-based techniques for remote monitoring and programming of robots [3-6]. At the same time, the XML files becoming increasingly popular in production management as a data exchange format [7-8]. The question then arises, whether the philosophy of XML can be applied to robotics in order to describe the robot's workspace, both for documentation purposes and further processing of data useful for programming task. We could use the XML document as a form of database, where all objects placed in the robot's workspace have its representation in the form of tags.

As an example, let’s create the structure of the adequate XML document. First of all we define the root object named “scene”. In the XML convention, it must contain opening tag and closing tag. The definition will look like in Figure 3. Repeating this task, we could define other objects located in the robot’s workspace, like:

- storages (buffers),
- machines,
- object of manipulation,
- places for reorientation of the manipulated object,
- tool storage station, etc.

```
<scene>
...
</scene>
```

Fig. 3. The definition of the root element named “scene”

In fact, there could be many different objects, which should be defined in order to create the complete scene description. For this reason, we can make certain simplification – the object could be defined as it is shown in Figure 4.

```
<object>
    <type>type_of_object</type>
    <coordinates>
        <x_coord>x</x_coord>
        <y_coord>y</y_coord>
        <z_coord>z</z_coord>
    </coordinates>
    <angles>
        <a_angle>alpha</a_angle>
        <b_angle>beta</b_angle>
        <g_angle>gamma</g_angle>
    </angles>
</object>
```

Fig.4. The definition of the “object” element, which could be used as a code template

In this example it has been assumed that Cartesian coordinate system will be used. The “object” element (parent) is described by the “type_of_object” value, which could be any name (like “out_buffer” or “lathe_1” etc.), which represents corresponding object in the workspace. Next, we use “coordinates” element (child) to define the XYZ point, where the effector must stop – it is the position of the tool center point of the manipulator in relation to global coordinate system of the robot. In the same way the angles are defined, which are the ones between axes of the global and the effector’s local coordinate system – accordingly for x, y and z axis. It could be said that we orientate the object in space from the robot’s point of view – more precisely by placing its tool center point in position, which allows the robot to interact with the object. In this way, we have complete information about arrangement of the objects in the robot’s workspace and about the position of the manipulation object. The database of all objects, presented in the form of the XML document has the human-readable form and could be processed using standard XML tools.

Conclusions

The markup languages are more readily used in robotics. Because of flexibility and public availability of document creation guidelines XML language have gained the popularity and it is often used wherever data exchange and processing is required.

This paper presents the idea that is the part of larger work, which aim is to simplify the programming process of the robot, while providing the portability of a code. At the current stage, it cannot be treated as the complete solution, but only as the draft, because in the paper only overall structure was presented, without some elements that make the XML document valid (complied with the rules). Using the well-documented XML, it needs to refine the presented idea to meet the language standard requirements. Further works will focus on the preparation of the functional form of robot's workspace description, which could be processed in order to obtain data for programming the robot.

References

[1] C.F. Goldfarb, The Roots of SGML -- A Personal Recollection, [online] Retrieved from: http://www.sgmlsource.com/history/roots.htm [Accessed: 25 Jun 2013].

[2] XML - Managing Data Exchange, (Wikibooks, The Free Textbook Project), 2012, [online] Retrieved June 25, 2013 from http://en.wikibooks.org/w/index.php?title=XML_-_Managing_Data_Exchange.

[3] M. Makatchev, S.K. Tso, Human-robot interface using agents communicating in an XML-based markup language, Robot and Human Interactive Communication, 2000, RO-MAN 2000, Proceedings of the 9th IEEE International Workshop on Robot and Human Interactive Communication, Osaka, Japan, 2000, 270-275.

[4] Y. Zhang, A. Golovinsky, M. Yim, C. Eldershaw, An XML-based scripting language for chain-type modular robotic systems, 8th Conference on Intelligent Autonomous Systems (IAS), 2004.

[5] Q. Wang, S. Liu, Z. Wang, A New Internet Architecture for Robot Remote Control, Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, 2006, 4989-4993.

[6] G.G de Rivera, R. Ribalda, J. Colas, J. Garrido, A generic software platform for controlling collaborative robotic system using XML-RPC, Advanced Intelligent Mechatronics, Proceedings, 2005 IEEE/ASME International Conference on, 2005, 1336-1341.

[7] D. Krenczyk, Example of application of automatic generation models method for simulation systems (in Polish), Wybrane Problemy Inżynierskie, Politechnika Śląska, Gliwice, 2 (2011) 229-234.

[8] D. Krenczyk, Journal of Machine Engineering, 11 (2011) 171-180.