THE HOLONIC MODEL OF TASKS PERFORMED BY INDUSTRIAL ROBOTS IN TERMS OF THE MARKUP LANGUAGE

Krzysztof Foit

Silesian University of Technology, Institute of Engineering Processes Automation and Integrated Manufacturing Systems, Konarskiego 18A St., 44-100, Gliwice, Poland

Corresponding author: Krzysztof Foit: krzysztof.foit@polsl.pl

Abstract: The paper discuss the problem of representing the description of tasks performed by robot in the way that is both human-intelligible and machine-processable. This issue is important from the point of view of the organization of work of a technological process engineer and a robot programmer – such method could be seen as a tool that allows to describe the work performed by a robot or a group of robots and facilitates the creation of an appropriate program. This matter is not simple to solve, because robots' manufacturers use different specifications, programming languages and robots' manipulators have different construction. In order to overcome the difficulties and allow the further machine processing, the markup language has been used. Moreover, the holons philosophy has been used as the way of dividing the sequence of activities performed by the robot into smaller, coherent parts.
Key words: Markup language, XML, industrial robot, holon, pseudocode.

1. INTRODUCTION

Modern industry is characterized by dynamic changes in the context of evolving standards, market requirements and customer preferences. These features translate directly into the need to ensure manufacturing flexibility and implementation of new standards – including the Industry 4.0 philosophy. In this dynamic environment, each machine exchanges information with its surroundings and with other machines, acting according to the algorithm that optimizes the manufacturing process, reacts to unexpected events and maintains the process continuity. All algorithms are reflected in the source code of the machine control software. A general description of the actions performed by the machine can be presented in the form of a block diagram, but the graphical form of the description has a significant influence on detailedness. In most cases, the machines are controlled by a PLC and the standardization of the source code makes it easier to understand how the program works. The situation is different in the case of robots, where programming languages (and thus also source codes) may differ significantly from each other – in fact, each manufacturer uses its own solutions that are basically incomprehensible to a person unfamiliar with programming. The aim of this paper is to propose the more universal and understandable notation for the description of the tasks performed by the robots. The form of the description should be simplified and may use the terms that are similar to the natural language. On the other hand, the syntax should be machine-processable. One of the options that satisfy the mentioned conditions is the use of markup language.
The paper also raises the subject of using the holon methodology in order to organize the process of creating a description of the robot's activities. As the result, the outline of the new methodology will be presented.

2. RESEARCH MOTIVATION

Industrial robots are very specific machines among the others used in the manufacturing process. First of all, they are characterized by great versatility of the tasks performed – from transport to simple machining operations. For this purpose, various tools are used, whose operation and control differ in terms of complexity. Another aspect that complicates the modelling of tasks realized by a robot is the lack of standards in terms of robot's programming languages. In fact, each manufacturer uses its own implementation, which causes great difficulty in interpreting the source code unambiguously. For this reason, many scientists are working on alternative methods of robot programming, such as Programming by Demonstration or Task Level Programming (Billard et al., 2008), (Calinon, 2009), (Coste-Mainere et al., 1992), (Somani et al., 2016), (Brunete et al., 2016). These systems, however, in most cases have not gone beyond the walls of laboratories and still are not a useful methods of robot programming. They use both a descriptive language similar to natural (Task Level Programming) and various ways of "showing" the machine how to perform the task. One of the most interesting methods is to use gestures and speech to create a robot program (Biggs and MacDonald, 2003), (Voyles and Khosla, 1999), (Tsarouchi et al., 2016).

Appropriate systems recognize and interpret messages transmitted by the operator to the programming system. On this basis, a sequence of manipulator movements is created.

Literature analysis indicates that work on finding methods to simplify the robot programming process started about thirty years ago and continues to this day. Despite the significant progress in the field of IT, the presented methods have not yet become standards. Therefore, this paper proposes a slightly different approach to the problem. The basic assumption is to try to standardize the description of tasks performed by robots, with a moderate degree of simplification. For this purpose, markup language was used, however, assuming high flexibility in defining syntax elements, it was decided to use Extensible Markup Language (XML), which has the features mentioned above and a good support for machine processing. In addition, because each task could be divided into single operations, the holonic approach could be used simultaneously. Such synergic approach has a potential for building a computer system, dedicated for development of the source code of robots' programs.

3. TOOLS AND METHODS

In the next part of the paper the methods will be presented, the use of which is essential for developing a new way of describing the tasks performed by the robot.

3.1 The XML meta-language

The XML is the abbreviation for Extensible Markup Language. In order to explain what the XML is, the idea of the markup language should be presented first.

First of all, it should be stated that markup language is not a programming language – it has descriptive form instead. Originally, the markup language was used for text processing purposes, but today is widely used for storing information – it is used by different computer program as a format of files for saving databases, spreadsheets, documents, drawings etc. Because of the function of a tag, the markup language may be of presentational, procedural or descriptive type. For example, one can use the asterisk (*) symbol to indicate that the text between tags (e.g. "**text**") should be interpreted as written in bold. This is the example of procedural type, because tags informs the processor what to do with the text. On the other hand, we could use the "*<quote>*" tag to indicate that the text is cited from the other source. It could be seen that such tag may have both the procedural and descriptive character – it carries human-readable information and at the same time, during machine processing, the tagged text can be accordingly formatted. In general, the border between the descriptive and procedural marking is blurred. As it was mentioned earlier, a lot of modern software uses some kind of descriptive, markup-oriented file formats (like XML) to store the information, while during opening, such file is processed procedurally.

The XML markup language has mainly descriptive character. In this case, it is possible to define any tags' names, so it is possible to use words from everyday language. In this case, important is the fact that the name of the tag is not relevant from the processor point of view, but it is relevant from the semantic point of view, when the code should be understood by human. The XML is also the form of meta-language, what means that it could be used for creation of another language. The standard is developed and maintained by World Wide Web Consortium (W3C). In fact, the XML-based document descriptions become default formats for Microsoft Office or Open/LibreOffice files as well as for many other applications.

For the purposes of this study, the XML specification is only the basis for the development of a new concept for the description of the tasks performed by the robot.

3.2 The holonic approach

The idea of the holon was introduced by Arthur Koestler, who attempted to present an unified theory of physical systems. His aim was to explain self-organizational behaviour of such systems. The word "holon" originates from the Greek language and means an element that has a dual nature – something that forms a whole, but at the same time being a part of a larger whole (Koestler, 2013). A holon is intelligent, self-organizing part of structure that may be triggered by the signal sent from the higher level of hierarchy. The triggered holon acts on the basis of internal patterns of operation and does not require external control.

Holons can form structures, which are called holarchies: a holarchy represents the connections and dependencies between holons. The single holon can dynamically join or leave a holarchy without abandoning its internal independence – moreover, it is possible that one holon is the member of more than one holarchy. Although the holarchy is the arrangement of holons (Mella, 2009), some sources claim that externally such structure can be also seen as a holon (Suárez et al., 2013).

In the research presented in this paper, the holonic approach has been used as a tool to systematize the tasks performed by the robot. As a result – using a meta-language – the description of these activities is possible at different levels of detail, taking into account the repetitiveness of certain actions.

4. THE CURRENT STATE OF THE PROBLEM

Contemporary research in robotics often focuses on the problem of robot programming. Analysing literature sources, it can be seen that the XML is quite commonly used in robotics-related problems. In most cases, the use of XML syntax refers to files saved by applications and used to exchange information between programs. It happens, however, that this language is used indirectly (after translation) or directly (in the control application) to control the robot. Gauss et al. suggested using XML to remotely control an industrial robot. They also point out that this approach should work regardless of the brand or type of robot (Gauss et al.,2003). Makatchev and Tso propose to use XML as a form of programming language for the robot (Makatchev and Tso, 2000). Lutovac et al. have a similar idea, but in their paper they suggest translating the source code from the original L-IRL source code (Lola Industrial Robot Language) into XML code as more universal and portable (Lutovac et al., 2012).
In terms of the use of the holonic approach in robotics, it should be noted that most researchers focus on analysing or modelling systems in which a robot (or robots) is only one of the elements – most often the considerations relate to the so-called Holonic Manufacturing Systems. There are not many papers that directly concerns the description of robot activities or programming in the context of the use of holons and holarchy. Jarvis et al. carried out an analysis of the tasks in the robot cell, but the holonic model was developed for all machines involved in the technological process (Jarvis et al., 2005). The idea is therefore similar to the one presented in this paper, but the authors do not address the problem of the holarchic model of the robot’s program source, which will be discussed later in this publication.

5. THE REPRESENTATION OF THE ROBOT’S PROGRAM PSEUDOCODE USING THE XML

The pseudocode is informal, non-standardized language used for verbal description of an algorithm. The syntax of the pseudocode is not regulated in any way, but in most cases it is similar to high level languages like Pascal, C, BASIC, etc. The characteristic feature of pseudocode is that it does not contain many standard elements, which can be found in classical programming languages, such as variable declarations, compiler directives etc. Functions and procedures are often replaced by verbal descriptions without going into detail. However, other features of programming languages have been retained, such as calls of procedures and functions, conditional instructions and jumps.
The XML is a meta-language, so it is perfect for defining new descriptive languages, including pseudocode. On the other hand, as it was mentioned earlier, XML is not a programming language. For this reason, it is not possible to define conditional instructions directly. Lutovac et al. propose to solve this problem by introducing new language elements, in the form of *if...then...else* tags (Lutovac et al., 2012). This solution seems sufficient at the level of XML translation into another source code, but the way the file is interpreted by the parser depends only on the author of the solution, so it cannot be considered that the proposed approach is exclusively the one that is correct.
For most popular “*pick and place*” tasks, a simple pseudocode syntax is sufficient, especially at a high level of generality. The problem arises only when it comes to describe precisely how to operate the manipulator and tool (e.g. gripper). For example, let's analyse a simple task case where a robot moves objects from the input buffer to the output buffer by executing three cycles and then moves back to the neutral position, waiting for further commands. The schematic representation of the task is shown in Figure 1.

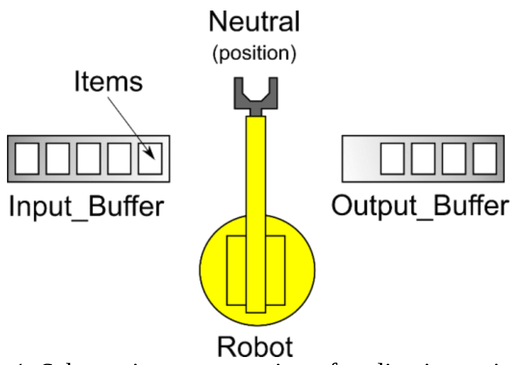

Fig. 1. Schematic representation of realization a simple “pick and place” task

The mentioned task has been described using pseudocode. In the Figure 2, two form of description is shown: the Pascal-like listing (Figure 2(a)) and the XML version of the code (Figure 2(b)). It could be seen that at this level of complexity, both codes are not complicated – the Pascal syntax is even more comprehensible, but the XML has a significant advantage, because it has the capability of machine processing of the code. This example also illustrates the use of *for* loop. This kind of loop is natively implemented in Pascal, so there is no problem to realize the fragment of code in question. The situation is different in the case of XML. As a descriptive language, its syntax does not provide the direct support for the loops – it depends on the interpretation of the tag by parser software. For this reason, the “<*loop*>” tag used in the example has both the procedural and the descriptive character. This approach is possible because the XML-based pseudocode is not directly compiled or interpreted, and is used only as a starting point to develop the source code of the robot program.

A more difficult task is to implement the conditional code in the XML. In the case of tasks performed by a robot, it is often necessary to use conditional instructions in the form of *if...then*. For example, the process shown in Figure 3 consists in sorting the elements taken by the robot from the input buffer. After taking up, they are stored in an appropriate output buffer, according to the colour of the object. Assuming that there are "Red", "Green" and "Blue" elements, a definition similar to the "*case of*" instruction could be used. As in the case of the loop, it is not about implementing an executable piece of code, but only a description of how to interpret the conditions and use this knowledge in the source code of the robot program.

a)

```
Program 'Pick-and-place';
    begin
        for i:=1 to 3
            begin
                MoveTo(Input_Buffer);
                Pick (Item);
                MoveTo (Output_Buffer)
                Place (Item);
            end;
        MoveTo (Neutral);
    end.
```

b)

```
<program name="Pick-and-place">
    <loop start="1" end="3">
        <move_to> Input_Buffer </move_to>
        <pick> Item </pick>
        <move_to> Output_Buffer </move_to>
        <place> Item </place>
    </loop>
    <move_to> Neutral </move_to>
</program>
```

Fig. 2. The pseudocode description of a “pick and place” task: the Pascal-style syntax (a) and the XML-style syntax (b)

In addition to the conditional function, which allows storing the element in a suitable storage unit, a conditional loop has also been defined, which is responsible for taking the element from the input storage unit and starting the colour check procedure. This is repeated as long as there are elements in the input buffer (see Figure 4).

```
<program name="color-sort">
    <while in_buffer_empty="false">
        <move_to> Input_Buffer </move_to>
        <pick> Item </pick>
            <select_case>
                <case itemcolor="Red">
                    <move_to> Red_Buffer </move_to>
                    <put> Item </put>
                </case>
                <case itemcolor="Green">
                    <move_to> Green_Buffer </move_to>
                    <put> Item </put>
                </case>
                <case itemcolor="Blue">
                    <move_to> Blue_Buffer </move_to>
                    <put> Item </put>
                </case>
            </select_case>
    </while>
    <move_to> Neutral </move_to>
</program>
```

Fig. 4. The pseudocode description of a “colour sort” task written using the XML-style syntax (b)

The use of the XML notation to write a pseudocode describing the tasks performed by the robot, in most cases, does not cause much difficulty. Using the ability of defining tags in an appropriate way, it is possible to write conditional instructions or loops, and the way they are written depends mainly on the capabilities of the parser. Unfortunately, the first problems with writing pseudocode in XML-based form appear when the robot's programming language uses a procedural or object-oriented approach. It should be remembered that in a classical high-level programming language, the definition of a procedure or function is precisely specified by means of a keyword, the name of the procedure or function and the parameters that are passed on during the call and possibly returned after the execution. In the case of

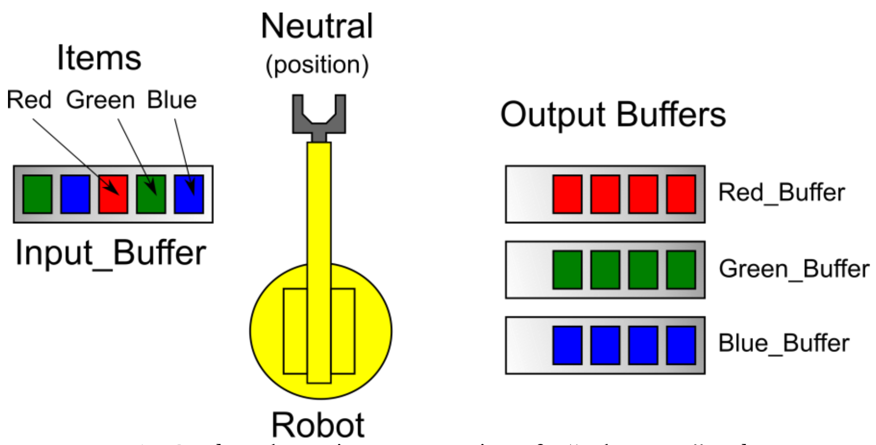

Fig. 3. The schematic representation of a “colour sort” task

the XML code, it is not impossible to define a procedure or function in the same way what makes the pseudocode less readable. The issue is further complicated by the fact of using different sets of parameters for the same functions, performed on different robots of different manufacturers. For example, it is possible to determine the speed of movement in specific physical units, give the number of the speed range or define as a fractional value, related to the maximum speed, expressed as a percentage, etc. If the definition refers only to a pseudocode, then some robot-specific properties can definitely be omitted, although the code remains at a higher level of generality. In the other cases, where the code will be machine-processed into an intermediate code or a ready-to-use source code, then particular accuracy must be kept when describing the implementation of individual elementary functions.

6. THE HOLONIC APPROACH TO THE ROBOT'S PROGRAM

The examples presented so far in the article were described on a high level of generality. In this way the description is similar to a natural language, where the communication is carried out with the use of the smallest number of details, which are necessary to pass the information without any errors. With regard to the description of the robot's activities, it can be said that the description at the highest level of generality is sufficient to understand the idea of the process – if necessary, the details could be clarified. Such approach allows describing very complicated tasks and avoiding the information overload simultaneously. Of course, it should be remembered that the pseudocode tags, used at a high level of generality "hide" a number of activities at a lower level. For example, the "<*pick*>" tag, which is connected with taking an object from the input buffer, defines an activity consisting of a series of movements and gripper control (open/closed) instructions.
In order to arrange the levels of details of the information, it is needed to define the concept of a holon in relation to the actions performed by the robot. Let the

$$H_{ij}(n) = \{I, P\} \tag{1}$$

will be the i,j holon of level n, where i is the number of the holon on the n-th level and j denotes the number of the holon on (n-1)-th level that is the superholon for the holon in question (equation (1)). In general, for further consideration, the holon H_{ij} is assumed to be a set of interfaces I and the data that describes the procedures P that are responsible for the operation of the holon. In the terms of programming language, the interfaces refer to the procedure/function calls and transferring the data from/to the caller. Similarly, the procedures part describes the actions realized by the holon.
According to the above definition, the superholon that reflects the highest level of generality, is placed on the level 0. The holons on the lower levels define the actions in more detailed way. As it was mentioned earlier, any task realized by the robot could be divided into smaller actions. Each activity can be further subdivided into sub-activities until no further decomposition is possible. The lowest level of the holons corresponds to the elementary activities. In this way, the program or its pseudocode transcription forms a holarchy, where the lower level holons (subholons) are activated by signals (call) from the superholons.
The holarchic approach to the structure of the robot's program, describing the actions performed by the robot, allows using the property resulting from the so-called Janus effect. This term refers to the beliefs of ancient Rome and the god Janus, who had two faces – one of them looked westwards, the other one directed eastwards. In the holon philosophy, Janus depicts the two-sided nature of holons: looking upwards, the holon is a part, while looking downwards could be seen as a whole. From the practical point of view it means that looking at the holon on the specific level, the lower levels holons could be hidden. For example, the procedure of picking the object ("<*pick*>" tag), could be defined as it is shown in Figure 5, but from the higher level, it is possible to see only the "*pick*" element.

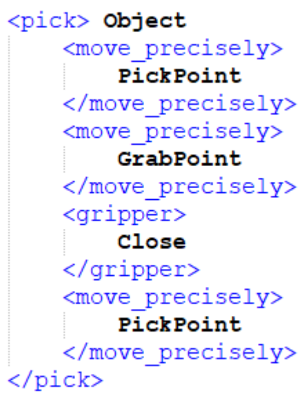
```
<pick> Object
    <move_precisely>
        PickPoint
    </move_precisely>
    <move_precisely>
        GrabPoint
    </move_precisely>
    <gripper>
        Close
    </gripper>
    <move_precisely>
        PickPoint
    </move_precisely>
</pick>
```

Fig. 5. The definition of the "*pick*" procedure

In this example, the procedure for gripper handling is the lowest level holon (basic). The "<*move_precisely*>" tag is reused three times – it is connected with the same holon, but different parameters are passed. The whole process is controlled by the "*pick*" holon.
The holonic approach can also simplify robot program creation. Because of the fact that a limited number of instructions are used to perform most of transport, assembly, or simple machining tasks, a description of the

robot's activities can be developed in the evolutionary way, starting with the most general description of the activities and then focusing on their elaboration until the level of elementary activities is reached.

7. CONCLUSIONS

The use of natural language to formulate tasks performed by robots is one of the contemporary trends in the development of robotics. Over a quarter of a century, many solutions have been developed to support the operator's work in the field of robot programming, but many of them have not left the walls of laboratories to become a tool, used on a daily basis in manufacturing plants. In this paper an attempt has been made to formulate the basis of the method, which will enable an evolutionary, multi-stage approach to the formulation of tasks carried out by an industrial robot. Starting with the simplest, most natural description of the task, the activity is further elaborated until the creation of the most detailed form of a pseudocode. For this purpose, the descriptive potential of the XML language was used, which was combined with a holonic approach to the problem of creating the code. Holarchic approach to the structure of the program and the separation of some parts of the program independently. At the same time, the use of top-down method minimizes the risk of making the mistakes in the early stages of task planning. In the context of further development of the method presented in the paper, it should be mentioned about the possibility of semi-automatic (supervised) creation of the descriptive pseudocode, using the self-organization property of holons.

8. REFERENCES

1. Billard, A., Calinon, S., Dillmann, R., Schaal, S., (2008). *Robot Programming by Demonstration*, Springer Handbook of Robotics, Siciliano B., Khatib O. (eds), pp. 1371-1394, Springer, Berlin, Heidelberg
2. Biggs, G., MacDonald, B., (2003). *A survey of robot programming systems*, Proceedings of the Australasian Conference on Robotics and Automation, Jonathan Roberts and Gordon Wyeth (eds), Available from www.araa.asn.au/conferences/acra-2003/table-of-contents/, Accessed 01/07/2019.
3. Brunete, A., Mateo, C., Gambao, E., Hernando, M., Koskinen, J., Ahola, J., Seppälä, T., Heikkila, T., (2016). *User-friendly task level programming based on an online walk-through teaching approach*, Ind Robot, **43**(2), 153-163.
4. Calinon, S., (2009). *Robot Programming by Demonstration*, EFPL Press.
5. Coste-Mainere, E., Espiau, B., Rutten, E., (1992). *A task-level robot programming language and its reactive execution*, Proceedings of 1992 IEEE International Conference on Robotics and Automation, pp. 2751–2756, IEEE, France (Nice).
6. Jarvis, J., Rönnquist, R., McFarlane, D., & Jain, L., (2005). *A team-based holonic approach to robotic assembly cell control*, J Netw Comput Appl., **29**(2), pp. 160–176.
7. Gauss, M., Buerkle, A., Laengle, T., Woern, J., Stelter, J., Ruhmkorf, S., and Middelmann, R., (2003). *Adaptive robot based visual inspection of complex parts*, Proceedings of the 34th International Symposium on Robotics, pp. 1–9.
8. Koestler, A., (2013). *Beyond Atomism and Holism - The Concept of the Holon*, The Rules of The Game: Cross-disciplinary Essays on Models in Scholarly Thought, Shanin, T. (ed.), pp. 233–247, Routledge, London.
9. Lutovac, M., Ferenc, G., Kvrgic, V., Vidakovic, J., Dimic, Z., (2012). *Robot programming system based on L-IRL programming language*, Acta Tech. Corviniensis, **5**(2), pp. 27-30.
10. Makatchev, M., Tso, S. K., (2000). *Human-robot interface using agents communicating in an XML-based markup language*, Proceedings of the 9th IEEE International Workshop on Robot and Human Interactive Communication, pp. 270-275, IEEE RO-MAN 2000, Osaka, Japan.
11. Mella, P., (2009). *The Holonic Revolution, Holons, Holarchies and Holonic Networks. The Ghost in the Production Machine*, Pavia University Press, Available from https://doi.org/10.13140/2.1.1954.5922, Accessed: 01/07/2019.
12. Somani, N., Rickert, M., Gaschler, A., Cai, C., Perzylo, A., Knoll, A., (2016). *Task level robot programming using prioritized non-linear inequality constraints*, Proceedings of 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 430–437, IEEE, Korea (Daejeon).
13. Suárez, S., Leitao, P., Adam, E., (2013). *Holonic recursiveness with multi-agent system technologies*, Trends in Practical Applications of Agents and Multiagent Systems, pp. 103–111, Springer, Cham.
14. Tsarouchi, P., Athanasatos, A., Makris, S., Chatzigeorgiou, X., Chryssolouris, G., (2016). *High level robot programming using body and hand gestures*, Procedia CIRP, **55**, pp. 1–5.
15. Voyles, R. M., Khosla, P. K., (1999). *Gesture-based programming: A preliminary demonstration*, Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), **1**, pp. 708–713, IEEE, USA (Detroit).

Received: July 25, 2019 / Accepted: December 15, 2019 / Paper available online: December 20, 2019 © International Journal of Modern Manufacturing Technologies