DATA COMPLETENESS VERIFICATION IN THE PROCESS OF PROVIDING THE INTEROPERABILITY OF PRODUCTION ENTERPRISE SYSTEMS

Jacek PĘKALA

Abstract: This paper presents the concept of checking the completeness of data after the process of its transformation during the exchange between various information systems used in production enterprises. The mechanism of conversion based on XSLT language is briefly characterized. Paper describes the problem of data loss during the conversion. Developed algorithm of data completeness verification is presented and the results of its implementation.

Keywords: data completeness verification, data transformation, data exchange, ERP, MES, interoperability, B2MML, XSL

1. Introduction

The development of information technologies allow you to create new solutions in many different areas or improve existing ones. In contemporary information structures, whose architecture is often created based on distributed solutions, using appropriate communication tools is essential [1]. This applies to manufacturing companies, which production activities take place often on large manufacturing plants and are subjected to the processes of automation and computerization. This results in a large amount of demanding top-down management or control information entities which are generating greater amounts of (often redundant) data [8]. Due to the specific operation of enterprise information systems, data exchange between them must be free of randomness, and the method of communication must be determined in advance. In the mechanisms for the exchange of information, in addition to the transmission of information, equally important is the role of the transformation, which is necessary because of differences even in the way they are stored. It is particularly important for the smooth flow of information between the two classes of systems occurring in the information structure of production enterprise: MES (Manufacturing Execution Systems) and ERP (Enterprise Resource Planning). In the flow of data between the systems B2MML language is used. This language specification has been built on the basis of the ISA-95 standard in compliance with the specification of XML. Responsible for the transformation is XSL language, closely related to the public standard XML. Despite having the right tools of transformation, data transformation is not flawless. Generally, this process causes the loss of information due to incorrectly defined rules of transition, or their lack. The purpose of this paper is to present the concept of a solution to check the completeness of the data undergoing the conversion process.

2. Data flow between enterprise production systems

2.1 Information structure in production enterprise

In all manufacturing companies, data acquisition and its management on production layer should serve to raise the efficiency and reliability of production. The acquisition is a key element in the decision making process and at every level of management in the company - from the operational services and maintenance, through the departments of engineering, to the administrative units.

These levels also have their reference in the hierarchical information structure of the company. At the lowest level there are sensors, actuators and a variety of industrial automation devices having a direct connection with the company's shop floor layer. Level above is domain of the Supervisory Control And Data Acquisition (SCADA), Computer Numerical Control (CNC), Programmable Logic Controller (PLC) and other industrial control systems. These systems operate in real time and despite of being responsible for data collection they have also control of machines and production lines components in their agenda[7]. In execution of production and high-level management areas in an industrial enterprise, there are two classes of systems used – Manufacturing Execution Systems (MES) and Enterprise Resource Planning (ERP), respectively. MES systems are responsible for the effective realization of the production process on the basis of accurate and up-to-date production data from lower-level systems. ERP domain is the management of the company which includes supply chain management, human resources, finance, etc. The above-mentioned systems are mutually complementary solutions and their possible interoperability adds value to the company. Their co-operation and the associated mutual communication is as important as any functionality that different systems provide, regardless of the presence of any other systems instances in the company's information structure. An important element of a modern MES system is the ability to seamlessly integrate with automation systems. To do that the common, open communication standards such as ISA-88 and OPC are used [3]. In the case of exchange of information from MES system with the parent ERP system is used ISA-95 standard and based on the XML-functional implementation – Business To Manufacturing Markup Language (B2MML). The exchange of information between those two class of systems (MES and ERP) is as important to the company as the data flow between the other levels. It is the subject of discussion of this paper, in particular, the analysis of the loss arising

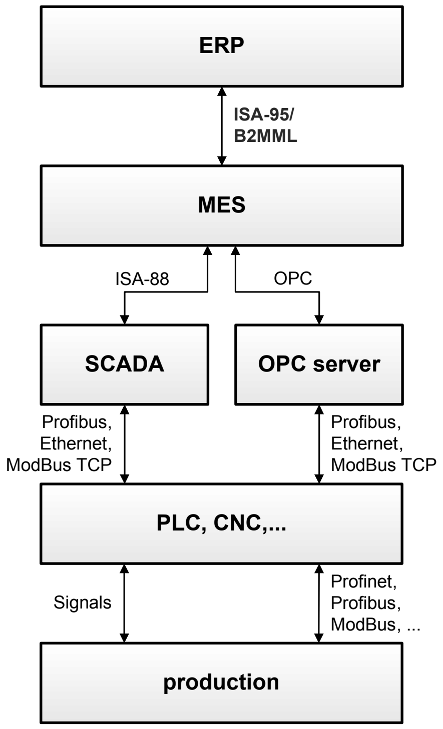

ERP
ISA-95/ B2MML
MES
ISA-88
OPC
SCADA
OPC server
Profibus, Ethernet, ModBus TCP
Profibus, Ethernet, ModBus TCP
PLC, CNC,...
Signals
Profinet, Profibus, ModBus, ...
production

Fig. 1. IT infrastructure model in an industrial enterprise

from the information. Figure 1 shows the hierarchical model of information systems in the industrial structure including communication standards.

2.2. ISA-95 standard and B2MML language

ANSI/ISA-95 Enterprise-Control System Integration is an international standard approved by a group of manufacturers, systems' providers and their contributors. It is described in several documents broadly understood systems integration methodology, which consists of five parts. If we assume that the ISA-95 standard presents a theory on the integration of management (ERP) and production (MES) systems, the B2MML language should be recognized as the executive arm of that standard. In [2] the author gives a simple definition B2MML language as XML-based implementation of the ANSI/ISA-95 standard. B2MML contains a set of XML schemas defined in the eXtensible Schema Definition (XSD) language, which include definitions of object-oriented models drawn from the content of the ISA-95 standard. The main goal for the language is to mediate in the process of integration by conversion of data and the structure of messages exchanged between the systems. The combination of XML and ISA-95 provides many tangible benefits in the process of information transfer. Besides its openness, simplicity and independence, XML schemas are easily adaptable to the needs of data exchange, which requires preserving the uniformity and consistency of the data structure. A significant advantage of XML and B2MML is the legibility of the information resulting from a clear structure. However, it's important to notice that the B2MML is not standard but an interpretation of standard and that the small details can be understood differently by different system vendors and users.

2.3. Style-sheet-based data transformation

EXtensible Stylesheet Language (XSL) in an XML-based transformation language of XML documents. It allows to translate documents from one format such as XML to any other format compatible with XML syntax, including the above-mentioned B2MML. With great simplicity, ease of implementation and widespread use of XML as a standard for information recording, XSL is a universal tool which applies to many types of software [9].

The input in the transformation process is the source XML document and XSL stylesheet that specifies an XML document transformation. A stylesheet is made up of templates. Each template describes how to convert a part of the input document to the output document fragment. These data are processed by an XSLT processor - an application that can interpret XSLT stylesheet and - basing on input - generate the output document. Execution of transformation is based on evoking a template matching a specific input element. Figure 2 shows a simplified diagram of the flow of information with inclusion of message transformation to B2MML intermediate format. In addition to the above description, it should be added that the document B2MML (like any XML document) is not a "flat" file. It has a tree structure, and the data stored in it are hierarchical. XSLT uses templates to the tree elements that match selected patterns, and therefore XSLT provides a set of rules describing the transformation of one XML tree to new one. The processor in the transformation process creates a new tree.

The mechanism of the XSLT processor work during the transformation process can be divided into two major parts. At first, XML document is prepared for transformation, and then transformation process is performed. As a preparatory step, parsing of XSLT stylesheet and the source XML document is primarily made. As a result of the parse their trees are created. Then, the excess whitespace are removed from documents, followed by the attachment of standard XSLT rules to the trees.

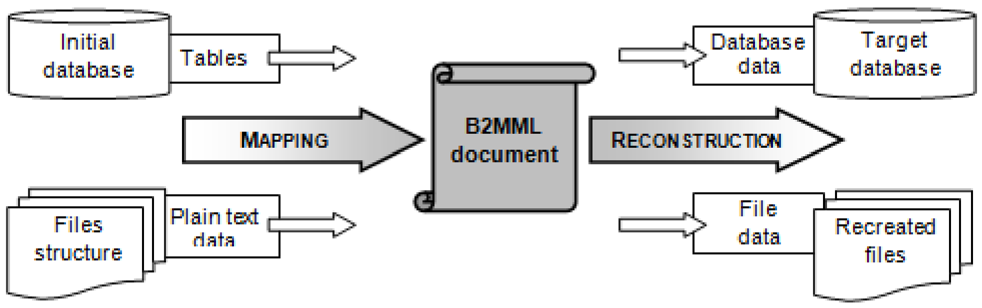


Fig. 2. Schema of the information flow between IT systems including the transformation process

After dissection of documents, processor goes to the core of the transformation. First, it creates a main element of the output tree, and then elements of the input tree from the root element are processed, and as a result is the output tree returned. Within the content of the output tree, each element of the input tree is processed according to standard procedure. As part of it, the best template to match the transformed node is sought. Of all the templates that match the processed element, the template with the highest priority is selected. These activities are repeated in a loop for each item in the source tree.

2.4. Inadequacies of data processing

The biggest drawback of transformations performed by XSL stylesheets is formation of incomplete output files stripped of the part of data from the input document. Some of the data from the source file is lost during the conversion. This is primarily the result of incorrectly defined XSL transformation files, which are devoid of template definition responsible for the conversion of specific information. In the absence of such a template, information is completely ignored by the XSLT processor and overlooked when creating a new data structure in the output file.

Another issue is the common problem of lack of certain information in the source file, which is expected by the target system. This is the consequence of specificity of certain system operation, which does not provide the information needed by other co-operating systems because it does not need it t work for itself. Stylesheets alone does not create non-existing data even if they have templates prepared for their conversion. Therefore, it is impossible to supplement an output document with a missing data during the transformation process. The question arises what lacks after conversion and what are the structural differences of the data between the source and output files. To answer this, a look at the structure of documents as well as particular transaction operations carried out in the conversion process is needed. When sending messages by input system to another dedicated system, they pass double transformation process. They go to the middleware layer that translates metadata and structure of messages into B2MML document (*schema conversion*), and then translates data from the B2MML format to the target figure. Metadata and structure information transformation into appropriate for target system requires not only a transformation of metadata but also its semantic parts, if necessary (*data* *conversion*). This situation can occur for example, when you change the format of a date, when one of the systems determines the order by *day-month-year* and another by *month-day-year*. Only after this treatment and obtaining a guarantee that a proper system will receive it, the document can be forwarded [6].

3. Data completeness verification

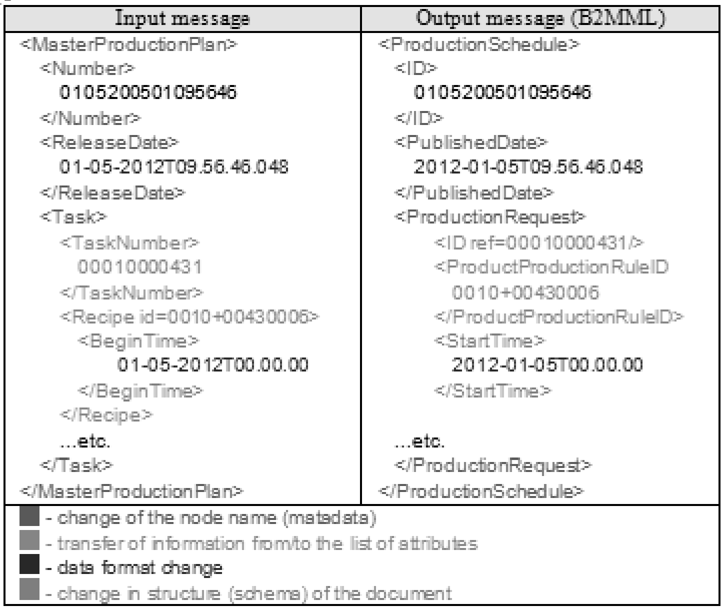
3.1. Classes of differences in data structure

Basing on the analysis of the data transformation process, stylesheets content and comparison of the contents of files before and after conversion, the four basic classes of changes made in the structure of documents and the resulting differences between them were determined by the author:

- renaming of nodes (metadata), including namespaces,
- changing of data format (node or attribute values),
- data transfer from or to a list of attributes of the particular nodes (due to lack of the existence of an attribute node or redundant),
- change in the structure (schema) of the document by moving subordination of particular nodes to other (related to the lack of appropriate node in the localization or the existence of redundant).

Types of changes in the following example from [5] are shown in Tab. 3.

Tab. 5. Comparison of document content before and after its transformation



Input message	Output message (B2MML)
<MasterProductionPlan>	<ProductionSchedule>
<Number>	<ID>
0105200501095646	0105200501095646
</Number>	</ID>
<ReleaseDate>	<PublishedDate>
01-05-2012T09.56.46.048	2012-01-05T09.56.46.048
</ReleaseDate>	</PublishedDate>
<Task>	<ProductionRequest>
<TaskNumber>	<ID ref=00010000431/>
00010000431	<ProductProductionRuleID
</TaskNumber>	0010+00430006
<Recipe id=0010+00430006>	</ProductProductionRuleID>
<BeginTime>	<StartTime>
01-05-2012T00.00.00	2012-01-05T00.00.00
</BeginTime>	</StartTime>
</Recipe>	
...etc.	...etc.
</Task>	</ProductionRequest>
</MasterProductionPlan>	</ProductionSchedule>

▩ - change of the node name (matadata)
▨ - transfer of information from/to the list of attributes
■ - data format change
▨ - change in structure (schema) of the document

3.2. Syntax analysis

Among the potential differences in the data structure special attention focuses a problem that constitutes the composition of two and even more of the basic types of differences listed in point. 3.1. Quite likely the situation can occur in which compared files will have important differences in the structure related with the node transfer in the structure to another location and alteration of its name and the format of the information it holds at the same time. An example of such a node is an element <*StartTime*> (formed from element <*BeginTime*>) located in B2MML message presented in the Table. 1, the value of which is given in two different forms (fomats).

The original form of the algorithm of data completeness verification dealt with each of the individual differences. Thus, if a node was not present in the output structure, or there were more of them than the original version, the algorithm detected this difference, reported it and proceeded to further tree elements. Thus, in the hypothetical situation in which one node is moved to a lower level and becomes subject to a different element, the algorithm reported incomplete data. In fact, this particular information was still contained in the message, but elsewhere. This problem would not be too difficult to detect if transferred elements had the same name, for each node from the source file is compared according to this criterion to nodes in a file that has undergone a process of transformation. The question therefore arises how to associate the two elements together, which are not only at different levels of the hierarchy of their documents, but also have other names, even though they store the same information. One solution can be inclusion of lexical dictionaries associating individual words and phrases with their synonyms and equivalents. Another option, proposed by the author is to parse the data and metadata. If similarities in syntax of tags (node names) are found, an assumption can be made that a particular value of the comparator element comes from the element to which it is compared.

Regardless of the check node name, a similar mechanism of textual analysis may be used in the case of data, the format of which can vary. Only by combining the two mechanisms of verification of the data completeness it can be determined if compared nodes exceed arbitrarily set the similarity threshold and therefore can be treated as identical or not. To achieve this objective, syntax analysis was based on the Levenshtein distance, which calculation algorithm belongs to the group of edit algorithms.

3.3. Levenshtein distance

Levenshtein distance was proposed by Vladimir Levenshtein in 1965, as a measure of dissimilarity words (finite strings) [4]. Informally, it is assumed that the Leveshtein distance between the two words is equal to the number of single-character modifications necessary to change one word to another. Distance is defined as the minimum number of changes needed to transform one string into another, and the modification can take place only through the operations of insertion, deletion or substitution of the character. Equation (1) presents the mathematical definition of the Levenshtein distance, where:

lev - means Levenshtein distance,

a, b - two compared strings,

i, j - length of strings a and b, respectively [4].

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} 0 & , i = j = 0 \\ i & , j = 0 \text{ i } i > 0 \\ j & , i = 0 \text{ i } j > 0 \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+[a_i \neq b_j] \end{cases} & ,else \end{cases} \tag{1}$$

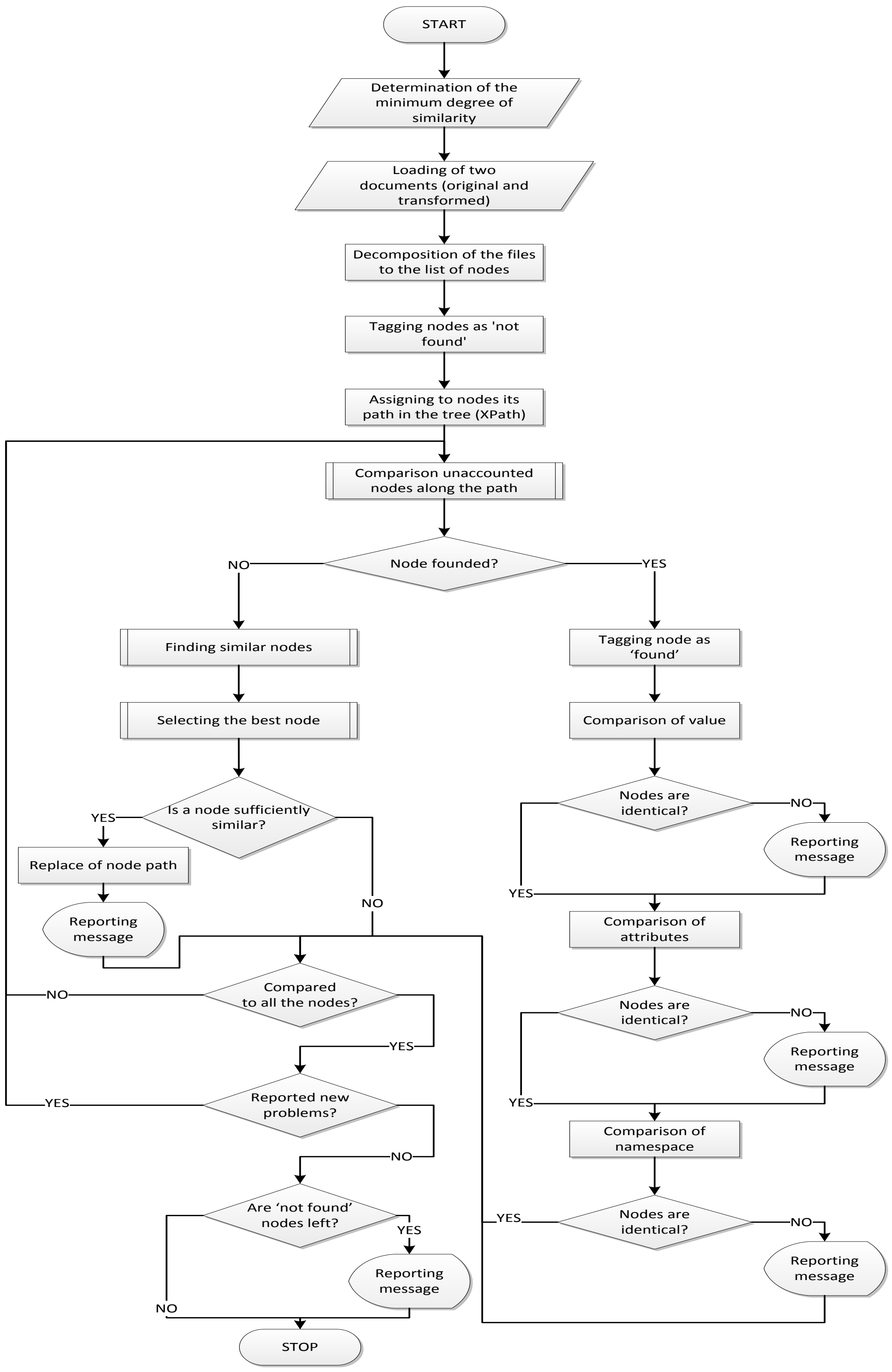


Fig. 3 Diagram of the data completeness verification algorithm

The value of Levenshtein distance for the example words: *MARK* and *ARIADA* is 5, because it takes at least five actions to transform the first word in the second: removing the letter *M*, switching letter *K* on *I* and adding three letters: *D* and *A* (two times).

3.4. Data completeness verification algorithm

Basing on the classification of the differences between files presented in point 3.1 and Levenshtein distance calculations discussed in point 3.3 used for syntax analysis, the algorithm has been developed to check the completeness of data (Fig. 3). It is designed to compare the contents of files before and after the transformation discussed in terms of types of differences and including parsing.

The mechanism operates according to the following procedure:

1. Initiating of algorithm.
2. Reading from interface a minimum degree of similarity required to define two different elements as similar.
3. Reading compared XML files.
4. Decomposition of loaded XML files to the list of nodes (two lists - the source and target elements).
5. Designation (flagging) all listed nodes as 'not found'.
6. Assigning to each node on the list its path in the tree (XPath).
7. Execution of the set of instructions on each ‘not found’ node from the source list, including:
    a. Verification whether a node in source list was found in the target list (checking whether the element on the target list has the same path in the tree).
    b. Selection among ‘not found’ nodes all 'similar' nodes to the comparator one. Default as 'similar' are treated all nodes that have the same name as the node sought.
    c. Selection of the node with the highest similarity among the ‘similar’ nodes. The similarity is defined as the sum of points given for:
        - the value of a text node (using the Levenshtein algorithm),
        - attributes,
        - the name of the element (using the Levenshtein algorithm),
        - namespace,
        - node’s child elements.
    d. Verification if the node selected as the most similar (the one that received the most points during the comparison) is sufficiently similar (exceeds the minimum similarity threshold specified in point 2).
    e. Replacement of the path of compared element to the path of its counterpart from the source list (this involves a change in the XPath for all node’s child elements) in the case of fulfillment of the condition set out in point. 7d.
    f. Reporting of finding node. In the list of target nodes, a node was found with the same path as the source node. Both nodes are marked as ‘found’.
8. After marking the node as a 'Found' the algorithm proceeds to perform a series of operations on the found elements of the products, including:
    a. comparison of the text nodes (without the text of the child nodes (children) if they are nested).
    b. verification if the text node values are equal.

c. if there is difference, notification message containing information on the Levenshtein distance between the compared values is presented.
d. comparison of all the attributes of nodes. It includes both locating source attributes in the target node, as well as its value comparison.
e. verification if the nodes have the same set of attributes. If the attributes are different (lack of attribute or redundant in the output node , different attribute values) the message is reported.
f. comparison of namespaces to which nodes belong.
g. verification if the nodes namespaces are identical. If the nodes belong to different namespace, the message is reported.

9. Verification if all nodes were compared (this checking allow to complete loop which is starting at point 7).
10. Verification if new problems have occurred after the loop completion.
11. Verification if the lists of nodes (source / target) have still 'not found' elements.
12. Reporting message for each 'not found' item in the list of nodes.
13. Stop.

By reporting notification message author meant exposure of the problem, which contains information on:

- class difference
- detailed information on the difference
- line of the document, in which the problem occur.

4. Conclusions

To pursue the purpose of this study an application was developed, which main task is comparing files in XML format, followed by determining the degree of completeness of the data compared to the reference document. The application has the ability to verify compliance with the specifications of an XML document, and the contents of its syntactic and semantic. The program contains simple code editor, which allows to edit documents. The application was written in C#, and to start it .NET Framework 4.0 installed is needed.

This solution has been tested on sample XML documents generated by the Wonderware MES system and files created by converting them to a format compatible with the B2MML specification. The results of the program are at the present stage of its development in line with expectations. This solution is part of the theory associated with the data exchange between ERP and MES as a middleware element which resides in environment of those systems. It allows to compare data from different backgrounds together and on the basis of differences between them to facilitate the completion of overlooked in the process of exchange information. It is an added value to the solution, which ultimate goal is interoperability at the management system level in a production enterprise. The current form of the algorithm can be developed for another methods of parsing.

Literature

1. Chwajoł G., The Evolution of Middleware used in distributed manufacturing control systems .Mechanika i informatyka, 2005, s. 18-20.
2. Gould L.: B2MML Explained. Automotive Design & Production, 2007, [www.autofieldguide.com/articles/b2mml-explained].
3. Klaus R., Stróżyk T.: Problemy integracji systemów IT zarządzania produkcją,

Komputerowo Zintegrowane Zarządzanie, Oficyna Wydawnicza PTZP, Opole 2012, ISBN 978-83-930399-0-6, tom I, s. 799-810
4. Levenshtein V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady, 10/1966, s. 707–710.
5. Pękala J.: Wykorzystanie języka B2MML jako narzędzia integracji informacji w przedsiębiorstwie produkcyjnym w ramach standardu ISA-95. Research Reports Project CII-SK-0030-03-0708, 2008, s. 304-309.
6. Pękala J., Gadzina K.: Transformacja danych z wykorzystaniem formatu B2MML jako element integracji systemów informatycznych przedsiębiorstwa. Pomiary, Automatyka, Robotyka, 2/2012, s. 151-156.
7. Skura K., Smalec Z.: Integracja systemów informatycznych w automatyzacji procesów produkcyjnych. Pomiary, Automatyka, Robotyka, 7-8/2005, s. 6-11.
8. Zając J., Chwajoł G.: Integracja informacji w systemach sterowania wytwarzaniem. PIAP AUTOMATION'2005, 2005, s. 96-105
9. XSL Transformations, [pl.wikipedia.org/wiki/XSL_Transformations].

Mgr inż. Jacek Pękala
Instytut Technologii Maszyn i Automatyzacji Produkcji
Politechnika Krakowska
31-864 Kraków, al. Jana Pawła II 37
tel./fax: (0-12) 374 32 26 / (0-12) 374 32 02
e-mail: pekala@m6.mech.pk.edu.pl