MECHANIKA
CZASOPISMO TECHNICZNE
TECHNICAL TRANSACTIONS
MECHANICS
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ
8-M/2012
ZESZYT 22
ROK 109
ISSUE 22
YEAR 109

JACEK PĘKALA*

KONCEPCJA ALGORYTMU SPRAWDZANIA KOMPLETNOŚCI DANYCH W PROCESIE ICH WYMIANY MIĘDZY SYSTEMAMI INFORMATYCZNYMI PRZEDSIĘBIORSTW PRODUKCYJNYCH

CONCEPT OF DATA COMPLETENESS VERIFICATION ALGORITHM IN THE PROCESS OF ITS EXCHANGE BETWEEN PRODUCTION ENTERPRISE SYSTEMS

Streszczenie

W artykule przedstawiono wstępną koncepcję schematu sprawdzania kompletności danych po procesie ich transformacji, opracowaną na potrzeby wymiany informacji między różnymi systemami informatycznymi przetwarzającymi informacje w przedsiębiorstwie produkcyjnym. Scharakteryzowano mechanizm konwersji informacji wykorzystywany podczas ich przepływu pomiędzy systemami informatycznymi przedsiębiorstwa. Opisano problematykę utraty części danych w trakcie ich przekształcania. Zaprezentowano wyniki działania programu implementującego opracowany algorytm.

Słowa kluczowe: algorytm sprawdzania kompletności danych, transformacja danych, wymiana danych

Abstract

The paper presents an initial version of a data completeness verification schema used in the process of its transformation elaborated for the information exchange between different systems in a manufacturing company. Data conversion mechanism used in information flow between enterprise systems was characterized. Problems of data loss during the conversion process were described. The paper also presents the work results of the application which implements the algorithm.

Keywords: data completeness verification algorithm, data transformation, data exchange

* Mgr inż. Jacek Pękala, Instytut Technologii Maszyn i Automatyzacji Produkcji, Wydział Mechaniczny, Politechnika Krakowska.

Oznaczenia

B2MML – *Business to Manufacturing Markup Language* – język znaczników relacji zarządzanie-produkcja
CNC – *Computer Numerical Control* – komputerowe sterowanie urządzeń numerycznych
ERP – *Enterprise Resource Planning* – system zarządzania zasobami przedsiębiorstwa
MES – *Manufacturing Execution System* – system realizacji produkcji
OPC – *OLE for Process Control* – łączenie i osadzanie obiektów dla kontroli procesu
PLC – *Programmable Logic Controller* – programowalny sterownik logiczny
SCADA – *Supervisory Control and Data Acquisition* – system kontroli nadzorczej i akwizycji danych
XML – *Extensible Markup Language* – rozszerzalny język znaczników
XSD – *Extensible Schema Definition* – rozszerzalna definicja schematów
XSL – *Extensible Stylesheet Language* – rozszerzalny język arkuszy stylów
XSLT – *Extensible Stylesheet Language Transformation* – przekształcenia oparte o rozszerzalny język arkuszy stylów

1. Wstęp

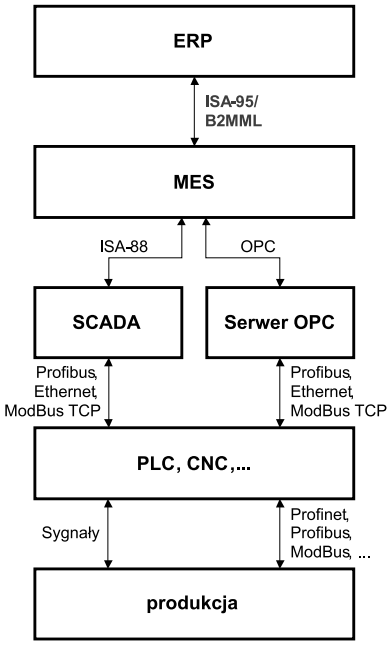
Rozwój technologii informatycznych pozwala na tworzenie w wielu różnych dziedzinach nowych rozwiązań czy usprawnienie już istniejących. We współczesnych strukturach informatycznych, których architektura jest często tworzona w oparciu o rozwiązania rozproszone, zastosowanie odpowiednich narzędzi komunikacyjnych jest niezbędne [1]. Dotyczy to w dużej mierze przedsiębiorstw produkcyjnych, których złożone działania wytwórcze, odbywające się nierzadko na wielkopowierzchniowych halach produkcyjnych, poddawane są procesom automatyzacji i komputeryzacji. Powoduje to powstawanie dużej ilości wymagających odgórnego zarządzania czy sterowania podmiotów informatycznych, generujących jeszcze większe ilości (często nadmiarowych) danych [7]. Ze względu na specyfikę pracy systemów informatycznych przedsiębiorstwa wymiana danych pomiędzy nimi musi być pozbawiona przypadkowości, a metody komunikacji muszą być z góry określone. W mechanizmach wymiany informacji, obok samego przesyłania informacji, równie istotną rolę pełni ich przekształcanie, które konieczne jest z racji różnic chociażby w sposobie ich przechowywania. Stanowi ono szczególnie ważny element sprawnego obiegu informacji między dwoma klasami systemów występujących w strukturze informatycznej przedsiębiorstwa produkcyjnego: MES i ERP. W obiegu danych pomiędzy tymi systemami wykorzystuje się język B2MML. Specyfikacja języka została zbudowana na bazie standardu ISA-95 przy zachowaniu zgodności ze specyfikacją języka XML. Za przebieg transformacji odpowiedzialny jest blisko związany z językiem XML ogólnodostępny standard XSL. Pomimo posiadania właściwych narzędzi transformacji, przekształcanie danych nie jest pozbawione wad. Z reguły proces ten powoduje utratę części informacji w wyniku błędnie zdefiniowanych reguł transformacji bądź ich braku.

Celem niniejszego artykułu jest przedstawienie koncepcji rozwiązania pozwalającego na sprawdzanie kompletności danych przechodzących proces konwersji. Wykorzystano do tego został ogólnodostępny język XML przeznaczony do zapisu różnych danych w strukturalizowany sposób.

2. Przepływ danych między systemami informatycznymi w zakładzie produkcyjnym

2.1. Struktura informatyczna przedsiębiorstwa

We wszystkich firmach produkcyjnych pozyskiwanie danych z warstwy produkcyjnej i zarządzanie tymi danymi służyć ma podnoszeniu wydajności i niezawodności produkcji. Akwizycja jest kluczowym elementem w procesie podejmowania decyzji i to na każdym szczeblu zarządzania w firmie – od służb operacyjnych i utrzymania ruchu przez wydziały inżynieryjne aż po jednostki administracyjne. Poziomy te mają również swoje odniesienie w zhierarchizowanej strukturze informatycznej przedsiębiorstwa. Na najniższym poziomie znajdują się czujniki, elementy wykonawcze oraz różne urządzenia automatyki przemysłowej, mające bezpośredni związek z warstwą produkcyjną zakładu. Poziom wyżej znajdują się systemy kontroli i akwizycji danych SCADA, przemysłowe układy sterowania CNC, PLC i inne. Systemy te funkcjonują w czasie rzeczywistym i oprócz gromadzenia danych odpowiadają za sterowanie maszyn i elementów linii produkcyjnych [5]. W obsłudze realizacji produkcji oraz obszarów zarządzania wysokiego szczebla w przedsiębiorstwie przemysłowym stosowane są dwie klasy systemów – odpowiednio MES i ERP. Systemy MES odpowiedzialne są za skuteczne prowadzenie procesu produkcyjnego na podstawie dokładnych i aktualnych danych produkcyjnych pochodzących z systemów niższego poziomu. Domeną systemu klasy ERP jest zarządzanie zasobami przedsiębiorstwa, w tym łańcuchami dostaw materiałów, zasobów ludzkich, finansów itp. Wymienione wyżej systemy są rozwiązaniami wzajemnie komplementarnymi, a ich ewentualna interoperacyjność stanowi wartość dodaną dla przedsiębiorstwa. Ich współpraca i związana z nią wzajemna komunikacja jest równie ważna jak każda funkcjonalność, którą poszczególne systemy zapewniają niezależnie od obecności innych podmiotów w strukturze informatycznej. Istotnym elementem nowoczesnego systemu MES jest możliwość prostej integracji z systemami automatyki przemysłowej. Wykorzystują przy tym powszechnie stosowane otwarte standardy komunikacyjne jak ISA-88 czy OPC. W przypadku wymiany informacji z nadrzędnym dla niego systemem ERP stosowany jest standard ISA-95 i powstała na bazie języka XML jego funkcjonalna implementacja – język B2MML. Wymiana informacji pomiędzy systemami klasy MES i ERP jest równie ważna dla przedsiębiorstwa jak przepływ danych między innymi poziomami. Stanowi ona przedmiot rozważania niniejszej pracy, a w szczególności analiza powstających podczas niej ubytków informacyjnych. Rys. 1 przedstawia model hierarchii systemów informatycznych w strukturze przemysłowej z uwzględnieniem standardów komunikacyjnych.



Rys. 1. Model struktury informatycznej w przedsiębiorstwie przemysłowym

Fig. 1. IT infrastructure model in an industrial enterprise

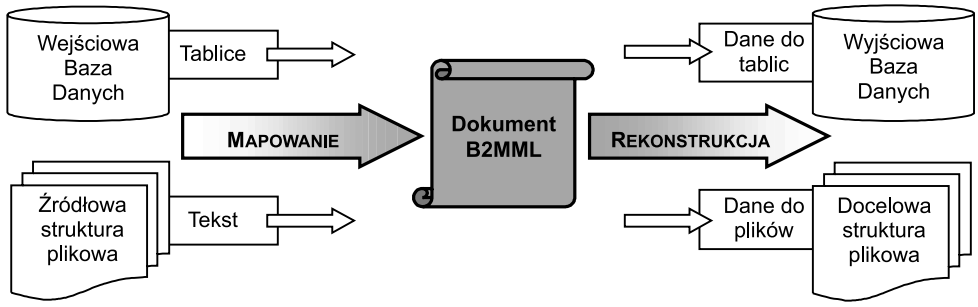
2.2. Standard ISA-95 i język B2MML

ANSI/ISA-95 Enterprise-Control System Integration to międzynarodowy standard zatwierdzony przez grupę producentów, dostawców systemów informatycznych i ich opiniodawców. To opisana w kilku dokumentach, składająca się z pięciu części metodyka szeroko pojętej integracji systemów. Standard nie przedstawia technicznego rozwiązania problemu, za to określa fundamenty pod jego realizację. Jeśli przyjąć, że standard ISA-95 prezentuje teorię dotyczącą integracji systemów zarządzania (ERP) i systemów odpowiedzialnych za realizację produkcji (MES), to za jego ramię wykonawcze uznać należy język B2MML (Business to Manufacturing Markup Language). W pracy [2] autor podaje prostą definicję języka B2MML jako opartą na języku XML implementację standardu ANSI/ISA-95. Język B2MML zawiera zbiór schematów XML zapisanych w języku XSD, w których zawarte są zaczerpnięte z treści standardu ISA-95 definicje modeli obiektowych. Celem nadrzędnym stawianym językowi jest pośredniczenie w procesie integracji systemów przez konwersję danych i struktury wiadomości przesyłanych między tymi systemami. Połączenie XML i ISA-95 przynosi wiele wymiernych korzyści w procesie transferu informacji. Poza otwartością, prostotą i niezależnością, schematy XML są łatwo adaptowalne do potrzeb wymiany danych, która wymaga zachowania jednolitości i spójności struktury danych. Znaczącą zaletą języków XML i B2MML jest czytelność informacji wynikająca z przejrzystej struktury. Należy jednak pamiętać, że język B2MML nie jest standardem, a pewną interpretacją standardu, która w drobnych szczegółach może być inaczej rozumiana przez różnych dostawców systemów i użytkowników.

2.3. Transformacja danych z wykorzystaniem arkuszy stylów

XSLT to oparty na XML-u język przekształceń dokumentów XML. Pozwala na przetłumaczenie dokumentów z jednego formatu XML m.in. na dowolny inny format zgodny ze składnią XML-a, w tym także na wspomniany już B2MML. Dzięki dużej prostocie, łatwości implementacji i powszechnemu stosowaniu XML-a jako standardu dla zapisu informacji, XSLT jest uniwersalnym narzędziem znajdującym zastosowanie w wielu rodzajach oprogramowania [6].

Danymi wejściowymi w procesie transformacji jest źródłowy dokument XML oraz arkusz stylów XSL, określający sposób transformacji dokumentu XML. Arkusz stylów składa się z szablonów. Każdy szablon opisuje, jak zamieniać pewien fragment dokumentu wejściowego na fragment dokumentu wyjściowego. Dane te przetwarzane są przez procesor XSLT – aplikację, która potrafi interpretować arkusz XSLT i na podstawie danych wejściowych wygenerować dokument wyjściowy. Wykonanie transformacji polega na wywołaniu szablonu pasującego do konkretnego elementu. Rys. 2 przedstawia uproszczony schemat przepływu informacji z uwzględnieniem transformacji wiadomości do międzyoperacyjnego formatu B2MML. W uzupełnieniu do powyższego opisu należy dodać, iż dokument B2MML (jak każdy dokument typu XML) nie jest plikiem „płaskim". Ma strukturę drzewa, a dane w nim przechowywane są zhierarchizowane. XSLT stosuje szablony do elementów drzewa pasujących do zadanych wzorców, a zatem XSLT zawiera zbiór reguł opisujących przekształcenie jednego drzewa XML w nowe. Procesor w procesie transformacji tworzy nowe drzewo.

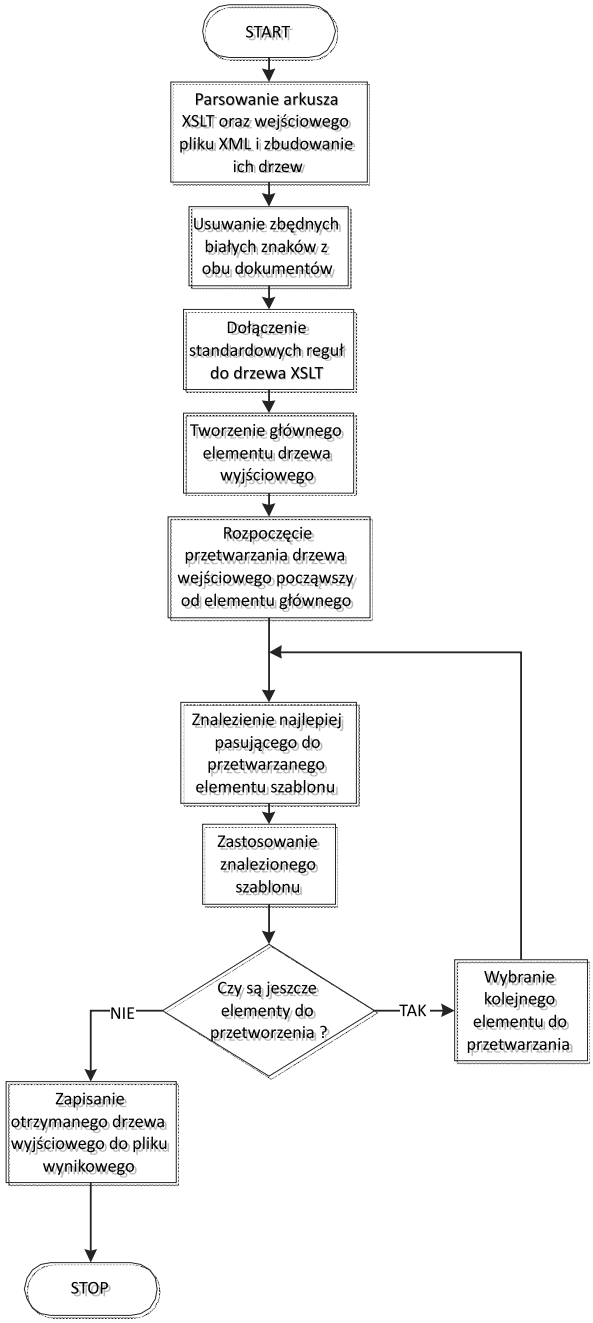


Rys. 2. Schemat przepływu informacji pomiędzy systemami z uwzględnieniem procesu transformacji

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

2.3.1. Algorytm transformacji

Mechanizm działania procesora XSLT (rys. 3) podczas procesu transformacji można podzielić na dwie zasadnicze części. W pierwszej dokument XML jest przygotowywany do przekształcenia, w drugiej wykonywana jest procedura transformacji. W kroku przygotowawczym dokonywane jest przede wszystkim parsowanie arkusza XSLT oraz źródłowego dokumentu XML. W wyniku parsowania utworzone zostają ich drzewa. Następnie z dokumentów usuwane są nadmiarowe białe znaki, a w dalszej kolejności do drzewa XSLT dołączane są standardowe reguły [6].



Rys. 3. Schemat blokowy algorytmu transformacji XSLT

Fig. 3. Block diagram of a XSLT transformation algorithm

Po spreparowaniu dokumentów procesor przechodzi do zasadniczej części transformacji. Najpierw tworzony jest główny element drzewa wyjściowego, a następnie przetwarzane są elementy drzewa wejściowego, począwszy od elementu głównego, w efekcie czego zwracane jest drzewo wyjściowe. W ramach utworzonego drzewa wyjściowego każdy element drzewa wejściowego przetwarzany jest następująco [6]:

– Znajdowany jest najlepiej pasujący szablon. Ze wszystkich szablonów pasujących do przetwarzanego elementu (każdy szablon nienazwany ma wzorzec – atrybut *match*) wybierany jest ten o najwyższym priorytecie (obliczonym na podstawie atrybutu *priority*, postaci wzorca oraz pozycji w dokumencie).
– Znaleziony szablon zostaje zastosowany. Elementy szablonu znajdujące się w przestrzeni nazw XSLT traktowane są jak instrukcje i odpowiednio interpretowane. Pozostała część jest kopiowana do drzewa wynikowego.
– Jeśli w szablonie umieszczona jest instrukcja *xsl:apply-templates*, procesor przechodzi w tym miejscu do rekurencyjnego przetwarzania listy elementów wskazanych atrybutem *select* lub – jeśli go brak – wszystkich potomków aktualnego elementu. Jeśli w szablonie brak jest instrukcji *xsl:apply-templates*, żadne z elementów aktualnego poddrzewa (dzieci i ich następniki) nie są w tym miejscu dopasowywane (przetwarzane). Mogą jednak zostać przeznaczone do dopasowania (za pomocą instrukcji *xsl:apply-templates*) z innego. Jeśli w szablonie umieszczona jest instrukcja *xsl:apply-templates*, procesor przechodzi w tym miejscu do rekurencyjnego przetwarzania listy elementów wskazanych atrybutem *select* lub – jeśli go brak – wszystkich następników aktualnego elementu. Jeśli w szablonie brak jest instrukcji *xsl:apply-templates*, żadne z elementów aktualnego poddrzewa (dzieci i ich następniki) nie są w tym miejscu dopasowywane (przetwarzane). Mogą jednak zostać przeznaczone do dopasowania (za pomocą instrukcji *xsl:apply-templates*) z innego szablonu [6].

2.4. Niedoskonałości przetwarzania danych

Największą wadą przekształceń dokonywanych za pomocą arkuszy stylów XSL jest powstawanie niekompletnych plików wynikowych, tzn. pozbawionych części danych znajdujących się w dokumencie wejściowym. Część danych z pliku źródłowego jest podczas konwersji tracona. Jest to przede wszystkim wynik źle zdefiniowanych plików przekształceniowych XSL, które pozbawione są definicji szablonów odpowiedzialnych za konwersję konkretnych informacji. W przypadku braku takiego szablonu informacje takie są całkowicie ignorowane przez procesor XSLT i pomijane przy tworzeniu nowej struktury danych w pliku wyjściowym.

Osobną kwestią pozostaje częsty problem braku pewnych informacji w pliku źródłowym, a których to system docelowy oczekuje. Wynika to bezpośrednio ze specyfiki pracy danego systemu, który informacji potrzebnych współpracującym z nim systemom nie obsługuje, bo sam ich nie potrzebuje. Arkusze stylów samodzielnie nie utworzą nieistniejących danych, nawet jeśli posiadają przygotowane do ich przekształcenia szablony. Nie istnieje zatem możliwość uzupełnienia dokumentu wynikowego w procesie transformacji. Pojawia się zatem pytanie, na ile przekształcone pliki są kompletne i jakie różnice w strukturze danych występują między plikami źródłowym i wynikowym. W tym celu należy przyjrzeć się strukturze dokumentów, a także poszczególnym operacjom dokonywanym na niej w procesie konwersji. Podczas wysyłania przez system wejściowy wiadomości dedykowanej innemu systemo-

wi przechodzi ona proces podwójnej transformacji. Najpierw bowiem trafia ona do warstwy oprogramowania pośredniczącego, która tłumaczy metadane i strukturę wiadomości na język B2MML (*schema conversion*), a następnie ponownie tłumaczy dane z formatu B2MML na postać docelową. Transformacja metadanych i struktury informacji na język właściwy dla systemu wyjściowego wymaga nie tylko przekształcenia metadanych, ale także jej części semantycznej, jeśli zachodzi taka konieczność (*data conversion*). Sytuacja taka może mieć miejsce np. przy zmianie formatu zapisu daty, kiedy jeden z systemów określa ją w porządku dzień-miesiąc-rok, a drugi miesiąc-dzień-rok. Dopiero po tym zabiegu i uzyskaniu gwarancji, że właściwy system go otrzyma, dokument może zostać przekazany dalej [4].

3. Sprawdzanie kompletności danych

3.1. Klasy różnic

Na podstawie analizy przebiegu procesu przekształcania danych, treści arkuszy stylów oraz porównania zawartości plików przed i po przekształceniu określone zostały przez autora cztery podstawowe klasy zmian dokonywanych w strukturze dokumentów i wynikających z nich różnic:

- zmiana nazwy węzłów (metadanych),
- zmiana formatu zapisu danych,
- przeniesienie danych z lub na listę atrybutów poszczególnych węzłów,
- zmiana struktury (schematu) dokumentu przez przeniesienie podległości poszczególnych węzłów na inne.

Typy zmian przedstawione na przykładzie z pracy [3] zostały przedstawione na rys. 4.

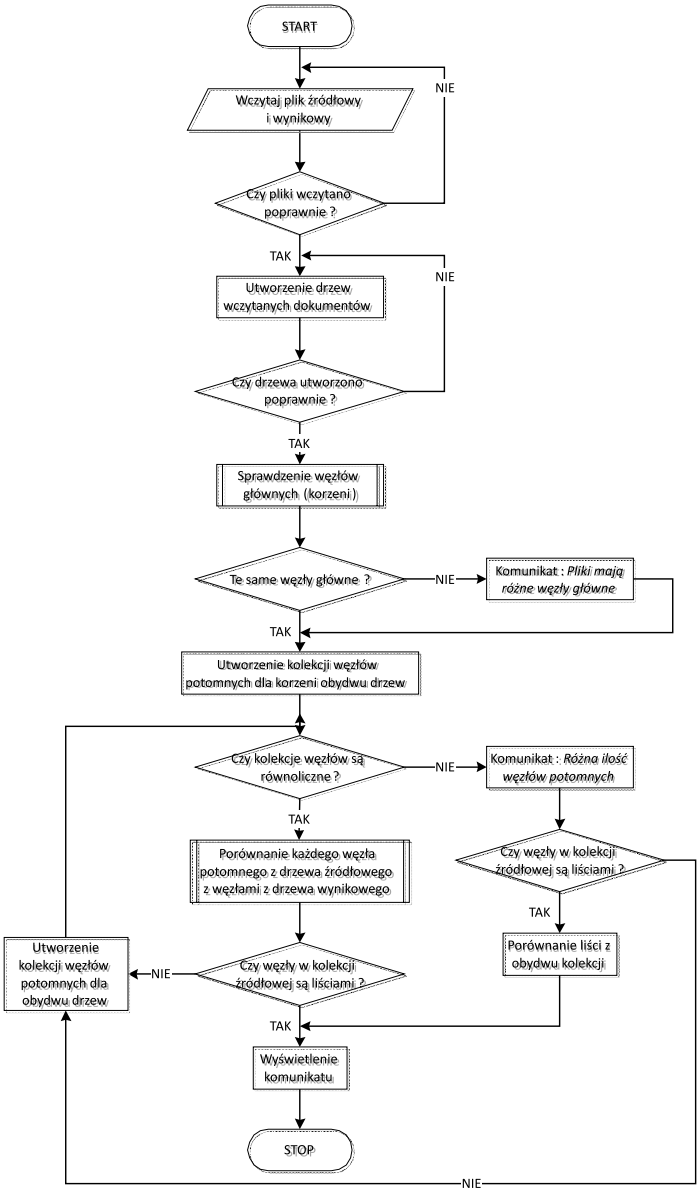
Wiadomość natywna:	Wiadomość wynikowa (B2MML):
<MasterProductionPlan> <Number> **0105200501095646** </Number> <ReleaseDate> **01-05-2012T09.56.46.048** </ReleaseDate> <Task> <TaskNumber> **00010000431** </TaskNumber> <Recipe id = **0010+00430006**> <BeginTime> **01-05-2012T00.00.00** </BeginTime> </Recipe>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

Rys. 4. Porównanie zawartości dokumentu przed i po jego transformacji

Fig. 4. Comparison of document content before and after its transformation

3.2. Algorytm sprawdzania kompletności danych

W oparciu o przedstawioną w pkt. 3.1 klasyfikację różnic pomiędzy plikami opracowana została wstępna postać algorytmu sprawdzania kompletności danych (rys. 5). Ma on za zadanie porównywać zawartość plików przed i po transformacji pod kątem omówionych klas różnic.



Rys. 5. Schemat blokowy algorytmu sprawdzania kompletności danych

Fig. 5. Diagram of data completeness verification algorithm

Mechanizm w pierwszej kolejności sprawdza poprawność wczytanych plików, które mają zostać porównane. W następnym kroku następuje parsowanie zawartości plików. W wyniku parsowania utworzone zostają ich drzewa. Po utworzeniu drzew algorytm przechodzi do etapu porównania węzłów głównych (korzeni) obu drzew oraz kolekcji ich węzłów potomnych. Wszystkie węzły potomne porównywane są z odpowiadającym mu węzłem w kolekcji węzłów z drzewa wynikowego. Każda para węzłów porównywana jest pod kątem każdego zdefiniowanego typu różnicy. Po tym etapie algorytm sprawdza, czy porównywane węzły nie posiadają własnych węzłów potomnych i dokonują ewentualnego porównania na nich. Jeśli węzły drzewa są liśćmi (nie posiadają podlegających węzłów), algorytm po ich porównaniu kończy pracę.

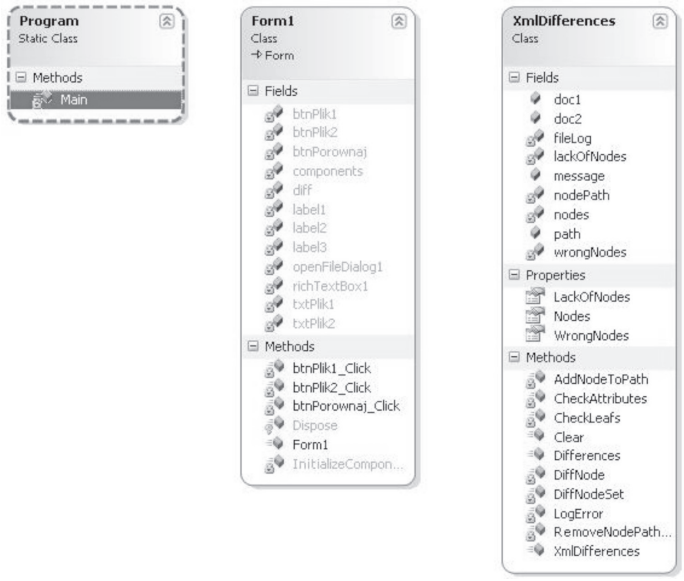
3.3. Implementacja algorytmu

Omawiany algorytm został zaimplementowany w stworzonej przez autora aplikacji porównującej zawartość dwóch plików. Aplikacja została napisana w środowisku Visual Studio 2008 w języku C#.

3.3.1. Opis aplikacji

Utworzone na potrzeby implementacji algorytmy i przedstawione na rys. 6 klasy to:

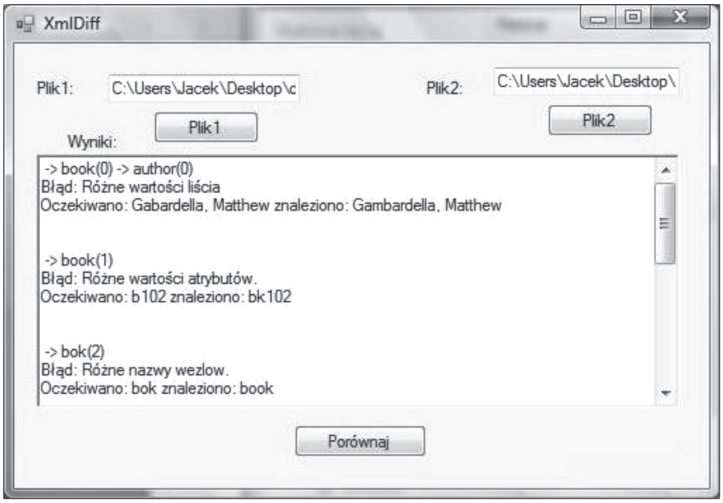
– *Program* – główna klasa programu, metoda Main() inicjalizuje parametry programu oraz tworzy formatkę Form1.
– *Form1* – formatka programu tworząca główne okno aplikacji, do której dodawane są pola tekstowe i przyciski.
– *XmlDifferences* – klasa implementująca algorytm porównania dwóch plików XML.



Rys. 6. Diagram klas

Fig. 6. Class diagram

Po uruchomieniu aplikacji pojawia się główne okno aplikacji, pozwalające na wprowadzenie za pomocą przycisków *Plik1* i *Plik2* ścieżki porównywanych plików XML. Przycisk *Porównaj* powoduje inicjalizację procedury porównywania zawartości wczytanych plików zgodnie z przedstawionym algorytmem. W efekcie jego działania wyświetlane są różnice w polu tekstowym *Wyniki*. Następuje także zapisanie wyników do pliku *log.log*. Rys. 7 przedstawia okno aplikacji z przykładowymi wynikami jej działania.



Rys. 7. Okno aplikacji wraz przykładowymi wynikami jej działania

Fig. 7. Application window with sample results of its work

3.2.2. Znaczenie i interpretacja wyników

Na rys. 6 przedstawiono przykładowe wyniki działania aplikacji. W prezentowanym przypadku zaobserwować można trzy różnice wykryte przez algorytm. W pierwszym z przypadków wystąpiła różnica w nazwie porównywanych węzłów, co spowodowało wyświetlenie następującego komunikatu:

-> book(0) -> author(0)
Błąd: Różne wartości liścia
Oczekiwano: Gabardella, Matthew znaleziono: Gambardella, Matthew

Pierwsza linia oznacza ścieżkę do węzła, w którym występują różnice. Liczby w nawiasie za nazwami węzłów wskazują ich kolejność w kolekcji węzłów tego samego poziomu (z zastrzeżeniem, że numerowanie elementów kolekcji rozpoczyna się od wartości 0). Druga linia podaje rodzaj błędu. Ostatnia linia podpowiada, czym różnią się porównywane węzły. W podobny sposób prezentowane są dalsze różnice występujące między porównywanymi plikami.

4. Podsumowanie

Prezentowane rozwiązanie zostało przetestowane na przykładowych dokumentach XML. Wyniki działania programu są na obecnym etapie jego rozwoju zgodne z oczekiwaniami. Rozwiązanie to wpisuje się w teorię związaną z wymianą danych między systemami klasy ERP i MES jako element warstwy pośredniczącej (*middleware*) pomiędzy systemami. Pozwala ona porównywać dane pochodzące z różnych środowisk ze sobą i na podstawie różnic między nimi ułatwić uzupełnianie pomijanych w procesie wymiany informacji. Stanowi ona wartość dodaną do rozwiązania, którego nadrzędnym celem jest interoperacyjność systemowa na szczeblu zarządczym w przedsiębiorstwie produkcyjnym. Należy jednak pamiętać, że jest to pierwsza postać algorytmu, który w dalszych pracach powinien ewoluować przez uwzględnienie chociażby analizy składniowej danych, co jasno wskazuje kierunek dalszego jego rozwoju.

Literatura

[1] Chwajoł G., *The Evolution of Middleware used in distributed manufacturing control systems*, III Ukraińsko-Polska Konferencja Młodych Naukowców „Mechanika i informatyka", 2005, 18-20.

[2] Gould L., *B2MML Explained*, Automotive Design & Production, 2007, www.autofieldguide.com/articles/b2mml-explained.

[3] 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, 304-309.

[4] Pękala J., Gadzina K., *Transformacja danych z wykorzystaniem formatu B2MML jako element integracji systemów informatycznych przedsiębiorstwa*, PAR, Pomiary, Automatyka, Robotyka Nr 2, Warszawa 2012, 151-156.

[5] Skura K., Smalec Z., *Integracja systemów informatycznych w automatyzacji procesów produkcyjnych*, PAR, Pomiary, Automatyka, Robotyka, Nr 7–8, Warszawa 2005, 6-11.

[6] *XSL Transformations*. pl.wikipedia.org/wiki/XSL_Transformations.

[7] Zając J., Chwajoł G., *Integracja informacji w systemach sterowania wytwarzaniem*, PIAP AUTOMATION'2005, 96-105.